TITLE Title Page

Systems Application Architecture
Common User Access
Basic Interface Design Guide

Document Number SC26-4583-00

COVER Book Cover

Systems Application Architecture

Common User Access
Basic Interface Design Guide

Document Number SC26-4583-00

EDITION Edition Notice
First Edition (December, 1989)

This edition applies to this release of *Systems Application Architecture, Common User Access: Basic Interface Design Guide*, SC26-4583, to IBM's Systems Application Architecture, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions.

Figures included in this document illustrate concepts that are presented and are not necessarily accurate in content, appearance, or specific behavior.

The implementation in specific IBM products of any new user interface functions or capabilities described in this book is subject to normal IBM business and technical reviews.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

International Business Machines Corporation
Department TJ7, Building 671
P. O. Box 60000
Cary, NC 27511

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

PREFACE Special Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM and others may have patents or pending patent applications, trademarks, or copyrights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, or copyrights, except as explicitly set forth herein. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States and other countries:

International Business Machines	IBM
Systems Application Architecture	SAA
Application System/400	AS/400
Operating System/400	OS/400
Operating System/2	OS/2
System/370	MVS/ESA
Presentation Manager	VM/XA

An asterisk (*) is used in the text of this book to identify the IBM trademarks.

CONTENTS Table of Contents

FIGURES Figures

TABLES Tables

FRONT_1 About This Book

This book provides a description of the Systems Application Architecture* (SAA*) Common User Access (CUA) basic interface for software applications developed to run on nonprogrammable terminals in the SAA* operating environments on System/370* and Application System/400* (AS/400*). The System/370* operating environments are MVS/ESA* TSO-E, VM/XA* CMS, VM/SP CMS, IMS/VS DC, and CICS/VS. The AS/400* operating environment is Operating System/400* (OS/400*).

Subtopics

FRONT_1.1 Who Should Use This Book

This book is written for designers and developers of software applications in the previously listed operating environments. The purpose of this book is to assist you in designing a user interface that is consistent within your application and across other applications. This book presents the user interface style guidelines and implementation considerations that you, the designer and/or developer, must be concerned with.

If you are specifically designing an application to run on the programmable workstation under *Operating System/2* Extended Edition* (OS/2* EE) or *Operating System/2* Standard Edition* (OS/2* SE), you should use *Systems Application Architecture, Common User Access: Advanced Interface Design Guide*, SC26-4582 as your guide.

This book describes user interface principles, techniques, and components that apply to a variety of software applications. In addition to describing what the components look like and how they work, this book gives guidelines on when to use them and what benefit they provide users.

This book also provides a basis for extending the components of CUA. In order to extend a component, it is important to understand how users benefit from it. It is also important to not change any of the basic elements of a component. You should always add to or extend, not supersede or change, the fundamental components in order to maintain consistency with and migration to other applications.

FRONT_1.2 How to Use This Book

Usually, the development of a good application starts with a set of goals, or *principles*, of the application. Based on these principles, a high-level design is created for the application, describing its overall function and style; this is the application *model*. Finally, there is the low-level design and implementation that is based on both the application principles and the model.

This book has been organized to work in conjunction with this application development process. It is intended to complement application development methods. Part 1, "Introduction and Principles" in topic 1.0 describes the principles for the CUA interface.

Based on these principles, Part 2, "Application Models" in topic 2.0, describes two interface models that can be implemented using programming tools in your software environment. The first model is the *text subset of the graphical model*. (The graphical model is described in *Systems Application Architecture, Common User Access: Advanced Interface Design Guide*, SC26-4582.) This interface style is referred to throughout this book as *text* or *text subset*. It is the type of interface described in *Systems Application Architecture, Common User Access: Panel Design and User Interaction*, SC26-4351. The other model is the *entry model*, which has most of the interface elements and interaction techniques contained in text, except for action bars and pull-downs.

Many applications originally developed to run on nonprogrammable terminals may be modified eventually to run on the programmable workstation under OS/2*. To support this migration, Part 2 also includes guidelines and considerations to help you make the transition smoother for you and your application users.

Part 3, "Component Descriptions" in topic 3.0 describes the fundamental components of the user interface.

User interface design is only one aspect of the development of usable software applications. Some other aspects are function, performance, supporting documentation, and national language support. Although the focus of this book is on the design and development of the user interface, you must not ignore the other aspects of application development because these aspects affect each other.

The figures in this book are *samples*. Each illustrates an implementation of panel layout rules and recommendations. Details of panel layout not specifically prescribed, such as the number of spaces between panel elements, are determined by application-enabling tools or the application designer.

FRONT_1.3 Systems Application Architecture

CUA is one of the four elements of SAA*, which is a collection of selected software interfaces, conventions, and protocols that collectively provide the framework for the development and execution of consistent applications. The other elements of SAA* are:

Common Programming Interface (CPI)

The CPI defines the languages and services that enable programmers to write application programs that can run in all SAA* environments.

Common Communications Support (CCS)

The CCS defines the architectures and protocols that interconnect SAA* systems and devices.

Common Applications.

Common applications are CUA-conforming applications that are written using the appropriate CPI interfaces and CCS functions.

Applications that are built using the main elements of SAA* enable both the developer and the users of the applications to be more productive. These applications also provide an increased level of consistency and connectivity across SAA* environments and enable an enterprise to make the best use of its resources.

FRONT_1.4 Related Publications

The following publication will help you understand the value and attributes of an SAA* application:

Systems Application Architecture, A Guide for Evaluating Applications, G320-9803.

The following publication discusses the definition of CUA for designers and developers of applications intended specifically for use on programmable workstations running OS/2*:

Systems Application Architecture, Common User Access: Advanced Interface Design Guide, SC26-4582.

This *Basic Guide* and the *Advanced Guide* replace the *Systems Application Architecture, Common User Access: Panel Design and User Interaction*, SC26-4351.

The following books contain the IBM rules, guidelines, and information necessary to provide national language support:

National Language Information and Design Guide, Volume 1: Designing Enabled Products, Rules and Guidelines, SE09-8001

National Language Information and Design Guide, Volume 2: Left-to-Right Languages and Double-Byte Character Set Languages, SE09-8002.

1.0 Part 1. Introduction and Principles

Subtopics
1.1 Chapter 1. Introduction to CUA
1.2 Chapter 2. Principles

1.1 Chapter 1. Introduction to CUA

The user interface is the means through which users communicate with the computer system. The Common User Access (CUA) is the definition of user interface components that should be the same across all applications. The definition is based on a set of generally accepted user interface design principles. It is optimized for ease of use and encourages learning by exploring. The fundamental objectives of CUA are:

- Usability and consistency within an application
- Consistency across applications.

When the framework, interaction techniques, and terminology are consistent across a variety of applications, users develop an expectation, or *conceptual model*, of how an application behaves. If new applications then support the same conceptual model, users can transfer what they have learned to other applications and predict how other applications will behave. This reduces the amount of time users need for learning new applications.

Computer users represent many diverse classes of users ranging from those who use *nonprogrammable terminals* attached to hosts to those who use personal computers that run either stand-alone or attached to hosts as *programmable workstations*. To balance the value of consistency between the two environments with the advanced capabilities of the programmable workstation, CUA defines two user interface models: *entry* and *graphical*. Where practical, similar user interface components are used across the models to enhance transfer of knowledge.

Subtopics

1.1.1 Entry Model

The *entry model* is the recommended model for the nonprogrammable terminal environment. It is best suited for applications that have panels with menus and prompts. It is intended for data-entry-intensive applications on nonprogrammable terminals and may also be appropriate for that type of application on the programmable workstation. The OS/400* operating system demonstrates many of the aspects of the entry model.

1.1.2 Graphical Model

The *graphical model* is recommended for individual applications, such as report writers and query products, designed to be used on a programmable workstation running with the OS/2* operating system. It makes extensive use of the windowing capabilities of that operating system. It includes action bars and their associated pull-downs, windows for messages, and parallel dialogs, such as help. It defines the use of standard graphical cues, such as check boxes. It is intended for all programmable workstation applications. The OS/2* user interface is an example of the graphical model.

1.1.3 Text Subset of the Graphical Model

The nonprogrammable terminal version of the graphical model is called the *text subset*. It maintains a close affinity to the graphical model, including action bars and their associated pull-downs and windows. It is intended for decision-intensive applications on nonprogrammable terminals. Applications developed with the ISPF dialog tag language can substantially conform to the text subset.

1.1.4 Workplace Environment

The *workplace* environment is an extension of the graphical model suitable for the integration of applications into an electronic version of a working environment that simulates a real workplace. For example, an electronic workplace for the office would have mail baskets, file cabinets, telephones, and printers that all applications could share. The separate applications are integrated as objects that appear as icons in the workplace environment. The workplace is optimized to the use of a mouse for moving objects on the screen. For example, to print a document, users use the mouse to *drag* an icon representing the document to an icon representing the printer. The workplace environment applies only to programmable workstations. The IBM* OfficeVision/2 is an example of the workplace environment.

1.1.5 CUA Documentation

These models are described in two SAA* documents. This book, the *Systems Application Architecture, Common User Access: Basic Interface Design Guide* is primarily for nonprogrammable terminal application designers. It describes the entry model and the text subset of the graphical model. It also describes techniques for migrating applications from the nonprogrammable terminal environment to the programmable workstation environment. The *Systems Application Architecture, Common User Access: Advanced Interface Design Guide* is primarily for OS/2* application designers. It describes the graphical model and the workplace environment.

1.1.6 CUA 1989 Versus CUA 1987

The first definition of CUA was contained in *Systems Application Architecture, Common User Access: Panel Design and User Interaction*, published in December, 1987. That book is being replaced by two SAA* books: this one, which is for the design of applications that run on nonprogrammable terminals and *Systems Application Architecture, Common User Access: Advanced Interface Design Guide*, which is for the design of applications that run on programmable workstations. The major change from the previous book is that the new design guides define the user interface models and contain increased consistency requirements.

The specific changes in the new design guides are:

Separation into two books, one for the programmable workstation and one for the nonprogrammable terminal.

This enables the application designer to concentrate on the interface best suited to the environment the application will run in.

Separation by interface model.

This enhances the clarity and usability of the information for application designers.

Selection of options based on model.

Guidelines are provided for each model that describe how to use the interface components to enhance consistency.

Inclusion of design principles and rationale for the user interface components that make up each model.

This enables application designers to extend the interface to meet the unique needs of an application while preserving consistency across applications.

Changes to some programmable workstation user interface components.

This enables exploitation of the graphical capabilities of programmable workstations.

Definition of additional user interface components.

This provides increased consistency.

1.1.7 Programming Tools

Applications that substantially conform to the entry model can be implemented with development tools for nonprogrammable terminals. These tools include ISPF and CSP for the MVS and VM environments, DDS for the OS/400* environment, BMS for the CICS/VS environment, and MFS for the IMS/VS DC environment. In addition, ISPF enables the development of the text subset of the graphical model on nonprogrammable terminals. The OS/2* Dialog Manager is intended to support the development of applications that conform to many aspects of the entry and graphical models for the programmable workstation.

1.1.8 Summary

CUA enables application developers to create applications that are usable and that have a user interface that is consistent across applications. The CUA definition spans a broad range of user requirements from an entry model optimized to nonprogrammable terminals to an integrated workplace environment optimized to programmable workstations. CUA represents the commitment of IBM to the user interface; CUA will continue to evolve consistently and to incorporate new technology. It is expected that the major evolution of CUA will take place in the programmable workstation, based on the integrated workplace environment.

1.2 Chapter 2. Principles

This section identifies design principles and describes techniques that support these principles. The principles and techniques are interrelated; therefore, the purpose for one technique may be similar to that of others. The definitions of user interface elements in the Common User Access are based on these principles and techniques.

It is important to understand these principles because the major portion of design is up to you. You must decide how to use and extend the components so they best support application-specific functions and provide the most usable interface. If you do not base an application on the design principles, you will find it difficult to produce a highly usable, consistent application; you may even find it difficult to use the interface components.

In general, there are two types of information presented to users: objects and actions.

Object is a general term for anything that users can manipulate as a single unit. Most objects are composed of sub-objects. For example, in a word processing application, the sub-objects of the document (the object) may be paragraphs, sentences, phrases, words, and letters.

Action is a general term for ways that users can modify, manipulate, or create objects. Actions modify the properties of an object or manipulate the object in some way. *Properties* are unique characteristics of an object; properties describe the object. In a word processing application, for example, the text in a document may have a type style property. Because the type style is a property, users can change it, for example, with a Set font action. Save and Print are examples of actions that manipulate an entire object.

Subtopics

1.2.1 Design Approaches

CUA describes two different design approaches for the application user interface: object-action and action-object.

An *object-action* oriented interface is one that first presents the objects with which users can work. *Objects*, not actions, are the focus of the users' attention. Users are expected to recognize and select one or more of the objects presented and then recognize and select one of the actions supported for the object or objects.

In this type of user interface, there is always a visual cue on the screen that indicates the set of objects with which users can work. An example of such a cue would be a list of the types and/or names of the objects, possibly with accompanying attributes or descriptions. In addition, the text subset also provides visual cues for available actions, so users can see which actions they can perform on the selected objects.

In contrast, an *action-object* oriented interface is one that first presents a list of actions. Users must first select an action to be performed and then choose the object or objects to perform the action on.

In some applications, the action-object orientation takes the form of a *command interface*, in which users must remember everything necessary to complete an action on an object. They must remember the name of the action, the name of the object, and the syntax of the command that will give the desired result. There are no visual cues to help users identify available objects or applicable actions. This type of interface is best provided in conjunction with a more visual interface, as a fast path for experienced users.

Often, the action-object orientation appears in the form of a hierarchy of menus. The first menu may be a list of the types of actions available to users. Users make a selection from that menu and progress to the next menu in the hierarchy, which might be a finer definition of the action type chosen. At the lowest-level menu in the hierarchy, users see a list of individual actions. Once they choose an action, they are asked to identify the object or objects on which the action is to be performed.

If a carefully designed application makes a limited number of actions available for each object and does not require users to understand its objects and their relationships, the action-object approach may be appropriate. The interface can provide visual cues to possible actions and available objects. It can be very easy to use, and may even be *walk-up-and-use*, because it leads users through a simple series of tightly controlled steps to complete a task.

For an application with numerous available actions and/or an application whose users need to understand the objects and their relationships, the object-action orientation is recommended. Some of the major reasons for this recommendation are:

- The object-action orientation avoids what are often referred to as *action modes*, which force users to switch from one mode to another to perform different actions on the same object.

- In a large system or an application with many actions, the action-object approach may require clustering of actions into higher levels, or categories, of action types. This often results in very large, complicated menu hierarchies. With the object-action approach, only the actions applicable to the users' chosen objects are presented. This avoids the need for large menu hierarchies.

- The object-action orientation combined with the visual approach used in CUA supports user exploration and user control, because only the actions applicable to selected objects are available to users. Users can view all available actions and perform them in the desired order, rather than in an order dictated by the application.

- In the action-object orientation, actions that apply to multiple objects may have to be given slightly different names, for example, synonym verbs, to distinguish their uses. Users may be confused by the similar action names or be led to believe that more actions are available for a given object than there actually are. With the object-action orientation, the same action name can be used for all objects to which it applies. The application knows how to tailor the action to the chosen object; users, therefore, do not have to distinguish among synonyms.

The larger the number of possible actions, the more difficult it is to refine and nest action types and names consistently and unambiguously. The general categories into which finer distinctions and actions are grouped may not be very meaningful in themselves. They often exist only for the purpose of grouping and may be arbitrary, abstract, or redundant. The object-action orientation, as noted previously, allows the application to offer only the actions applicable to the users' chosen objects. Hierarchies of action types and actions can, therefore, be minimized.

Appendix C, "Designing an Object-Action Oriented Application" in topic APPENDIX1.3 contains specific suggestions and guidelines for designing object-action oriented applications.

1.2.2 Design Principles

The two design principles are:

Users can develop a *conceptual model* of the interface.

Users can develop a conceptual model of how an application should work. The user interface should confirm the conceptual model by providing the outcome users expect for any action. This occurs only when the application model is the same as the users' conceptual model.

Users should be in control of the dialog.

You should allow users to control the dialog. Putting users in control means users should be able to perform any action in any sequence they want to complete their tasks.

Traditional applications that are sequential in nature do not support this principle. The object-action design approach supports this principle far better than the action-object approach. However, thoughtfully designed action-object applications can go a long way toward putting users in control. For guidance on when the action-object design approach is appropriate, see Chapter 4, " Entry Model" in topic 2.6.

1.2.3 Developing the Users' Conceptual Model

There are several techniques that develop and consistently reinforce the users' conceptual model of the user interface. They are:

- Using metaphors
- Designing a user-driven interface
- Making the user interface consistent
- Avoiding modes
- Making the interface transparent.

Subtopics

1.2.3.1 Using Metaphors

A *metaphor*, or analogy, relates two otherwise unrelated things. Writers use metaphors to help readers create a conceptual image, or model, of the subject. This same principle applies to applications. Using metaphors is a way to develop the users' conceptual model of an application. When you use metaphors that are familiar and real-world based, users can transfer previous knowledge of their environment to your application interface. You must be careful in choosing a metaphor to make sure the metaphor meets the expectations users have because of their real-world experience. Often, an application design is based on a single metaphor. For example, billing, insurance, inventory, and banking applications can present forms that are visually equivalent to the paper forms that users are accustomed to seeing.

1.2.3.2 Designing a User-Driven Interface

Another technique to develop and reinforce the users' conceptual model is to provide a user-driven design. Often the goal of an application is to automate what was a *paper process*. As users begin to use a computer to do their work, you should try to make their transfer simple and natural. Design your application so it allows users to apply their previous real-world knowledge of the paper process to the application interface. Your design can then support the users' environment and goals. As an important part of your application design, perform a task analysis to learn what users want to do and how they want to do it. For example, you should identify the audience for your application and study how they currently perform tasks. For more information, see Appendix D, "Recommended Readings" in topic APPENDIX1.4. Use the results of the task analysis to model the user interface for your application. You should test your application to make sure it meets the audience requirements.

1.2.3.3 Making the User Interface Consistent

Making the user interface consistent is one way to develop and reinforce the users' conceptual model of applications. Applications should be consistent with their hardware and software environments and throughout the applications themselves.

Consistency throughout an application is supported by establishing:

Common presentation: what users see

Common interaction: how users interact with the components

Common process sequence: how users and the computer communicate with each other

Common actions: how similar actions are implemented in the same way.

Common Presentation: Users become familiar with interface components when the visual appearance of the components is consistent and, in some cases, when the location of the components is consistent. This book describes the specific presentation and location for several components of the user interface. For example, the panel title is a user interface element that is consistent in both appearance and in location. Entry fields, however, are consistent in appearance but not in location; they can appear anywhere in the work area.

Common Interaction: After users can recognize interface components, they can interact with these components. When you consistently support the interaction techniques associated with each component, users become familiar with these techniques.

Common Process Sequence: CUA supports two process sequences that correspond to the two design approaches discussed previously in this chapter: *object-action* and *action-object*. When your application consistently supports one process sequence, users become familiar with the way to interact with the application. They also learn how the application responds. For example, if your application consistently supports the object-action sequence, users know that they must first select an object. They also know that a selected object is indicated, but no action is taken until they request an action. After users learn this way of expecting the computer to respond, their conceptual model would not be supported if users selected an object and your application immediately performed an action.

Common Actions: CUA defines *actions* that have specific meanings. Common actions provide a language between users and the computer so users can understand the meaning and result of actions. CUA defines common actions and terminology to assist you in providing consistency. For example, when users press the F1 key or an engraved Help key, they are telling the computer that they want to get help for the specific area of the application they are working on.

1.2.3.4 Avoiding Modes

Users are in a mode whenever they must cancel or complete what they are doing before they can do something else or when the same action has different results in different situations. Modes force users to focus on the way an application works, instead of on the task they want to complete. Modes, therefore, interfere with the users' ability to use their conceptual model of how the application should work. It is not always possible to design a modeless application; however, you should make modes an exception and limit them to the smallest possible scope. Whenever users are in a mode, you should make it obvious by providing good visual cues. The method for ending the mode should be easy to learn and remember.

Some types of modes are recommended by CUA. They are:

Modal Dialogs

Sometimes an application needs information to continue, such as the name of a file into which users want to save something.

Exception Conditions

When an application or system experiences an exception condition, users must take corrective action in order to continue their work.

1.2.3.5 Making the Interface Transparent

Do not make users focus on the mechanics of an application. A good user interface does not bother users with mechanics. Users view computers as tools for completing tasks, as a car is a tool for getting from one place to another. Users should not have to know how an application works to complete a task, as they should not have to know how a car engine works to get from one place to another.

A goal of user interface design is to make the user interaction with the computer as simple and natural as possible.

Making the Interface Simple

Your user interface should be so simple that users are not aware of the tools and mechanisms that make the application work. As applications become more complicated, users must have a simple interface so they can learn new applications more easily. Today's car engines are so complex they have on-board computers and sophisticated electronics. However, the driver interface remains simple: drivers need only a steering wheel and the gas and brake pedals to operate a car. Drivers do not have to understand what is under the hood, or even be aware of it, in order to drive a car, because the driver interface is still simple. Applications should provide the same simplicity for users.

Making the Interface Natural

The user interface should be intuitive so users can anticipate what to do next by applying their previous knowledge of performing tasks without a computer. An application, therefore, should reflect a real-world model of the user goals and the tasks necessary to reach those goals. One way to provide an intuitive user interface is to use metaphors, as discussed previously in this section.

1.2.4 Allowing Users to Control the Dialog

The second design principle is that users want to be and should be in control of the dialog. Traditional application designs put the application in control of the dialog, regardless of whether or not that design approach was appropriate for users. Putting users in control of the dialog requires a design approach that is fundamentally different than the traditional approach.

Users are in control when they are able to switch from one activity to another, change their minds easily, and stop activities they no longer want to continue. Users should be able to cancel or suspend any time-consuming activity without causing disastrous results.

The techniques that put users in control are:

- Making the interface forgiving
- Making the interface visual
- Providing feedback.

The text subset of the graphical model, which requires the object-action orientation, supports this principle and its supporting techniques. The entry model can support them to some extent, though to a lesser degree than the text subset.

Subtopics
1.2.4.1 Making the Interface Forgiving
1.2.4.2 Making the Interface Visual
1.2.4.3 Providing Feedback

1.2.4.1 Making the Interface Forgiving

The user interface should be forgiving. User actions should be easily reversed. When users are in control, they should be able to explore without fear of causing an irreversible mistake. Because learn-by-exploring environments involve trial and error, users should be able to back up or *undo* their previous action. Actions that are destructive (that may cause the unexpected loss of the users' information) require a confirmation. Users should feel more comfortable with a computer when their mistakes do not cause serious or irreversible results.

To users, the *unexpected* loss of their information is the most frustrating and destructive application occurrence. When you allow users the opportunity to change their minds about an action that would destroy significant data, you provide a forgiving interface, even if you are unable to allow them to *undo* the action after it is completed.

1.2.4.2 Making the Interface Visual

You should make your interface highly visual so users can see, rather than recall, how to proceed. Both the presentation of interface components and the user interaction with the components should be visual. Whenever possible, provide users a list of items from which they can choose instead of making them remember valid choices. The user interface described in this book supports many components that are visual.

1.2.4.3 Providing Feedback

Provide feedback for user interactions, whenever possible. Users should never perform an action without receiving visual feedback, audible feedback, or both. For example, color, emphasis, and other presentation techniques show users which choices they can select, when a choice has been selected, and when a requested action has been completed.

2.0 Part 2. Application Models

This section describes the two Common User Access models for nonprogrammable terminals: the *entry model* and the *text subset of the graphical model*, or *text subset*. This section also describes the fundamental user interface components you must consistently support in implementing both models and how the models will look when moved into the OS/2* environment.

Each *model* describes how interface elements work together, not how they are implemented. This section includes only the elements that are fundamental or recommended for each interface model. Part 3, "Component Descriptions" in topic 3.0 describes the implementation of individual components, optional elements, and the optional variations of elements that are included in CUA to accommodate some programming tools. For a list of fundamental, recommended, and optional elements for each model, see "Component Requirements by Model" in topic 3.1.3.

If you want to extend the user interface described in this section (for example, if you want to use graphical equivalents of interface elements or if you want to support a mouse), implement the extensions as similarly as possible to their descriptions in *Systems Application Architecture, Common User Access: Advanced Interface Design Guide*, SC26-4582.

Subtopics

2.1 Consistency

As you design an application, you should remember that it will be one of many that users can run. Users may switch back and forth between applications frequently. To achieve consistency under these conditions, applications must look and act alike to some degree. In some cases, the components of the applications must even be identical.

Presentation of CUA elements should be consistent across all applications running on a terminal, but applications should take advantage of the capabilities of the particular type of terminal being used. Some CUA elements have different appearances on different terminals, based on variations in hardware capabilities. Figure 1, Figure 2, and Figure 3 show some of the differences that occur because of the different character sets that are available on various terminal types.

PICTURE 1

Figure 1. Terminal Using Invariant Character Set

PICTURE 2

Figure 2. Terminal Using an Alternate Character Set

PICTURE 3

Figure 3. Terminal Using Another Alternate Character Set

The major visual difference between the entry model and the text subset of the graphical model is that the text subset makes use of action bars and pull-downs, while the entry model does not. However, most other user interface elements described in CUA are common to both models. This provides a significant degree of user interface consistency between text subset and entry model applications, even though their overall appearance is different and their underlying designs may be entirely dissimilar.

2.2 Ease of Use

These interface models are optimized for *ease of use* rather than for ease of learning. Users learn by exploring the system after some initial training. Once users have learned a few basic things about the interface, such as how to move the cursor to a choice and select it and how to get help, they should feel free to explore and continue to learn about an application.

An interface optimized for ease of learning is often called *walk-up-and-use*. This type of interface requires that applications lead users step by step through their activities and present information frequently to tell users what to do next and how to do it. New users find this very helpful. However, once they know how to use the applications, the continued presentation of tutorial information, once so welcome, becomes an annoyance.

In designing your applications, you should consider carefully before choosing to put *ease-of-learning* features in the user interface. In general, these features should be frequent only in applications that you expect will be used primarily by inexperienced users or will be used only a few times. A good place for such an application might be an automated information booth in an airport or hotel lobby. The entry model is better suited than the text subset for providing ease-of-learning features.

2.3 User Migration

As users move from one application or environment to another, they require some learning. However, by following a CUA model when you design your applications, you can reduce the amount of learning users require during migration.

For example, users of an entry model application require some learning when moving from use of an application on a nonprogrammable terminal to use on a programmable workstation. Users need to learn a few things about the programmable workstation environment; for example, the cursor does not move freely over the entire screen.

However, users do not need to learn the differences between underlying functions in the nonprogrammable terminal and the OS/2* environments; they are shielded from those functions by the application. Also, users need only minimal knowledge of the windowing functions of OS/2*. Once users have learned the few differences they need to know, they should experience no significant difficulty in using the application in the new environment.

A similar ease of user migration should occur for a text subset application that is moved from a nonprogrammable terminal to a programmable workstation.

Users moving from an entry model application to a text subset application will need to become familiar with the action bar and pull-downs. Also, entry model applications *may be* action-object oriented, while text subset applications *must be* object-action oriented. (See Chapter 2, "Principles" in topic 1.2 for information about these two orientations.) Migration between action-object and object-action orientation requires that users change their conceptual model of the interface, which increases the amount of learning required. The amount of learning can be reduced, however, if the entry model application is object-action oriented, like the text subset application.

2.4 Aspects of User Interface

All software applications have four aspects that describe their user interface:

Presentation: what users see

Interaction: how users interact with the components

Common actions: how similar actions are implemented in the same way

Process sequence: how users and the computer communicate with each other.

This section explains the interface models in terms of these aspects.

2.5 Chapter 3. Text Subset of the Graphical Model

This chapter defines the *text subset of the graphical model.* The text subset supports and requires the object-action process sequence. You should follow this model if your application has more than one action available for each object and/or if the users need to understand the objects and their relationships.

Subtopics

2.5.1 Process Sequence

In the text subset, the process sequence is *object-action* oriented. This means that the users select an object and then select an action to apply to that object. When you consistently support an object-action process sequence, you reinforce the users' conceptual model of the user interface.

The object-action process sequence has advantages over the action-object sequence. Some of the benefits of the object-action process sequence are:

It provides a basis for users to explore the application through context-sensitive actions. Users can first select an object and then browse the actions of an application to see which ones apply to the selected object. Users perceive that they are in control because they can see which actions are valid prior to attempting to perform them.

It allows users to perform a series of actions on a selected object rather than to repeatedly select the object for each desired action. For example, in a word processing application, users usually select some text (an object) and then apply to it a number of styles (actions), such as, bold, italic, and underlined. Users find an interface inconvenient and repetitive if they have to select an object many times to request all the actions they want to perform on that object.

It reduces the complexity of the user interface and simplifies its overall architecture by reducing the levels of action hierarchies needed for the application.

2.5.2 Presentation

Presentation is the visual aspect of the interface; it is what users see on the screen. The main visual components of the user interface are:

The primary window, which occupies the entire screen

Pop-ups

Panels, which are presented in windows and pop-ups and contain information for users or data for users to work with

The cursor.

Some of the visual elements that are common to most text subset applications are illustrated in Figure 4. The space between the panel title separator and the message area is the *work area*.

PICTURE 4

Figure 4. Panel Elements. This layout illustrates a panel in a primary window.

Subtopics

2.5.2.1 Panels

Panels are presented in windows and in pop-ups. The concept of presenting a panel in a window is illustrated in Figure 5. Information and application objects are presented in the work area of a panel. Application actions are presented in the action bar or function key area of a panel. Every panel in the primary window of a text application has an action bar. Panels in pop-ups do not have action bars. The panel in the primary window is the main focal point for the users' work activity. Panels presented in pop-ups supplement the dialog that is occurring in the primary window.

Displayed in a panel are standard interface components that provide a consistent way to present information to users. When users become familiar with the interface, they can quickly identify components of the interface and feel comfortable when using new applications that support these components.

PICTURE 5

Figure 5. Panel Presentation in the Primary Window

The panel elements that each application must consistently support are the action bar, panel title, work area, message area, command area, and function key area. Following is a discussion about each of these elements.

Action Bar: Each panel in the primary window of your text application must have an action bar. The action bar is positioned horizontally at the top of each panel and contains the application actions. Each action bar choice *must* have an associated pull-down. An action bar with a pull-down visible is shown in Figure 6. Chapter 10, "Action Bar and Pull-Downs" in topic 3.5 describes action bars and pull-downs and their interactions. "Standard Action Bar Pull-Downs" in topic 2.5.4.1 describes standard action bar choices, their associated pull-downs, and the dialog pop-ups associated with some standard pull-down choices.

PICTURE 6

Figure 6. Action Bar with Pull-Down Visible

Figure 7 illustrates the flow of the user navigation in a panel with an action bar. Specific rules for user interaction can be found in "How Users Interact with the Action Bar and Pull-Downs" in topic 3.5.7. Users select an object in the work area, then move the cursor to the action bar. They select an action bar choice and press the Enter key, causing the pull-down for the selected choice to be presented. Users then select a choice from the pull-down and press the Enter key to initiate the desired action. If users must supply additional information so the application can complete the action, the application presents one or more dialog pop-ups that request information. When users have supplied all necessary information, the application performs the requested action and then returns the cursor to the work area in the primary window. As shown in Figure 7, users press the Enter key to move forward through the process and the Cancel key to back up in the process.

PICTURE 7

Figure 7. User Navigation in a Panel with an Action Bar

Panel Title: In a primary window, the panel title contains the application name and, optionally, the object name, if there is a one. In a pop-up, the panel title identifies the function of the panel.

Work Area: The work area is the space between the panel title separator and the message area. It is the users' workspace and the focus of their attention. The application objects and much of the application information are presented in the work area. If the panel is too large to be presented completely in the window, then all or part of the work area is scrollable, as shown in Figure 5.

The work area of the panel in the primary window is the electronic counterpart of *a piece of paper*. For some types of applications, such as

editors and word processors, the work area is presented completely blank. For other types of applications, the work area may be customized. For example, a form fill-in application might present a customized form that looks very much like its paper counterpart.

The work area of the panel in a help pop-up contains information to assist users in continuing their interaction with the application.

The work area may contain a variety of interface elements. Unless noted otherwise, more information on these elements can be found in Chapter 7, "Panel Elements" in topic 3.2.

Scrolling information

Scrolling information is a visual cue to users that more information is available but is not currently visible and that the information can be brought into view using the scrolling keys. Scrolling information also tells users the direction in which the additional information is available. The **List of Policies** pop-up in Figure 8 shows an example of scrolling information.

PICTURE 8

Figure 8. Pop-Up Containing a Multiple-Choice Selection List

Selection fields and selection lists

Selection fields and selection lists are sets of related choices. They may be either single-choice or multiple-choice. Selection fields are not scrollable and contain a fixed set of choices. Selection lists typically vary in content or number of choices and are potentially scrollable. Figure 9 shows two single-choice selection fields, **Status** and **Type of order**. The pop-up in Figure 8 shows an example of a multiple-choice selection list. For more information about selection fields and selection lists and the users' interactions with them, see "Selection Elements" in topic 3.3.2.

PICTURE 9

Figure 9. Panel with Single-Choice Selection Fields and an Entry Field

Entry fields

Entry fields are spaces into which users type information. In Figure 9, the **Patient name** field is an entry field. For more information about entry fields and the users' interactions with them, see " Entry Fields" in topic 3.3.1.

Protected text

Protected text is information presented by the application that users cannot modify. Protected text is formatted in a way that is suitable for the data being presented. In Figure 8, the date and time display (**1/07/89 at 12:12**) is an example of protected text.

Headings and field prompts

Headings and field prompts identify entry fields, selection fields, selection lists, protected text, or related groups of those elements. Your application should provide one of these types of identifiers for each of these elements or each group of them, unless there is only one element or group in the panel and the panel title is sufficient as an identifier.

Column headings identify columns of entry fields, selection fields, selection lists, or protected text when all the items in the column are the same type. Figure 22 in topic 2.5.4.2.1 contains two column headings: **Objects** and **Last Updated**.

Group headings identify related groups of entry fields, selection fields, and protected text. Figure 10 contains three group headings: **Security Control, Printers**, and **Communication**.

Field prompts identify selection fields, entry fields, and variable output information. To guide users' eyes from a field prompt to its

associated field or item, use *leader dots* between the prompt and the field. Figure 10 illustrates field prompts with leader dots.

PICTURE 10

Figure 10. Panel with Group Headings and Field Prompts

Panel area separators

Panel area separators distinguish the action bar from the rest of the panel and distinguish adjoining areas within the work area from each other. The separator between the action bar and work area is a solid or dashed line. The separator between areas within the work area may be a solid or dashed line or a blank line. Because the message area is usually empty, it also may be used as a separator.

Message Area: Your application must provide a message area and locate it immediately above the command area. If there is no command area, locate the message area immediately above the function key area. Messages located in the message area should be removed when the panel is processed.

CUA defines three types of messages:

Information
Warning
Action.

When you need to provide a means within the message for users to respond to the message (for example, by providing a selection or entry field), or when it is especially important to get the users' attention, the message is displayed in a message pop-up.

Command Area: If your application has a command interface, users issue commands through the command area. You may provide the command area in either of two places:

In the primary window immediately above the function key area

In a pop-up, if you need to maximize the available space in the primary window and if you expect commands only to be used occasionally.

The command area begins with the following field prompt:

```
 Command ===>
```

Your application should provide a command interface if you want to allow users to issue system commands without leaving the application.

Your application also may allow users to type *application* commands into the command area. The command area can provide a valuable fast path for experienced users who prefer to enter an entire command at once, rather than issuing it by using the action bar. It is not necessary to provide command support if you want to allow users to perform only application actions, because those actions are provided through the action bar.

If the application allows users to perform application actions through both the action bar and the command area, the application should support those actions in the same way through both the command area and the action bar. The name of each command should be consistent with the name of the corresponding action in the action bar pull-down.

Function Key Area: The function key area appears at the bottom of the panel to present common actions and application-defined actions that users can request by pressing function keys. Users request actions listed in the function key area by pressing the assigned keys. The actions occur immediately.

2.5.2.2 Copyright Notice

If your application is protected by a copyright, the copyright notice should be displayed the first time users start the application. The copyright notice should be displayed in the message area of the first panel in the primary window or as protected text in the work area of that panel. Figure 11 illustrates an example of a copyright notice in the message area.

PICTURE 11

Figure 11. Example of a Copyright Notice

Your application should support the *About...* choice in the Help pull-down. The copyright notice appears in the pop-up that results from selection of the About... choice. For more information about the Help pull-down, the About... choice, and copyright information, see Chapter 15, "Help" in topic 3.10 and Chapter 17, "Copyright Information" in topic 3.12.

2.5.2.3 Pop-Ups

Pop-ups are bordered areas of the screen that supplement the dialog that is occurring in the primary window. The recommended appearance for pop-up borders is shown in the figures in this section. CUA defines four types of pop-ups:

- Dialog
- Prompt list
- Message
- Help.

The optional support for commands in pop-ups is discussed in Chapter 11, "Command Area" in topic 3.6.

The first three types of pop-ups are *modal*. That is, users must complete the pop-up dialog or cancel it before continuing interaction with the underlying panel.

Subtopics

2.5.2.3.1 Dialog Pop-Up

Through a *dialog pop-up*, users provide information needed to complete a dialog in the underlying panel. A dialog pop-up is, therefore, directly associated with the panel in the primary window.

PICTURE 12

Figure 12. Dialog Pop-Up. This ***Print Options*** dialog pop-up appears when users select the Print action in the ***File*** pull-down and press the Enter key.

2.5.2.3.2 Prompt List Pop-Up

A *prompt list pop-up* is presented when users request the Prompt action for an entry field that supports the Prompt function. The prompt list pop-up contains valid values for the entry field. When users select from the list in the pop-up and press the Enter key, the users' choice or choices are placed into the field.

Usability is enhanced by the Prompt action because

Users need only recognize the desired value.

The potential for errors in typing the value is eliminated.

PICTURE 13

Figure 13. Prompt List Pop-Up

2.5.2.3.3 Message Pop-Ups

A *message pop-up* is used to display a message when you need to provide a means within the message for users to respond to it (for example, by providing selection or entry fields), or when it is especially important to get the users' attention. Figure 14 shows an example of a message pop-up.

PICTURE 14

Figure 14. Message Pop-Up

2.5.2.3.4 Help Pop-Ups

Help pop-ups should be *modeless*. That is, users can continue to interact with other parts of the application while a Help pop-up is displayed. A Help pop-up is shown in Figure 15.

PICTURE 15

Figure 15. Help Pop-Up

2.5.2.4 User Options

CUA recommends support for the following *user* options, to help users tailor the application to their preferences and needs.

- Color per panel element type
- Display of panel IDs on/off
- Function key area display on/off
- Set beep on/off.

"Component Requirements by Model" in topic 3.1.3 shows the recommended user options and their default settings.

2.5.2.5 Application Use of Color

The use of color in an application might improve the presentation. An application should be visually appealing to users. To be effective, however, color must be used sparingly. Color should be used to support the exchange of information between users and the computer, not to have one piece of information compete with another.

You might confuse users by using too many colors, thereby diminishing usability. Overuse of color does not help users learn about the format and contents of the window nor does it help them to use the format and contents in the dialog. On the contrary, overuse of color challenges users to search for meaningful relationships between presented information--relationships that might not exist.

Following are design considerations for the use of color in interface design:

Use the same color for similar functions. For example, on a color nonprogrammable terminal, white is used for choices that are available for selection. Blue is used for unavailable choices. On a monochrome nonprogrammable terminal, available choices are presented in high intensity while unavailable choices are in low intensity.

However, the same color may be used to denote a different meaning for dissimilar functions if users will not misinterpret the multiple use of the same color. For example, blue is used not only for unavailable choices on a color nonprogrammable terminal but also for the action bar separator line and for pull-down borders. All three uses of the color blue are visual cues to users, but the uses do not conflict.

Avoid using color when other identification techniques are available, such as location. For example, the consistent location of the action bar and its pull-downs is easily learned. It is unnecessary, therefore, to uniquely color-code action bars and pull-downs; this would be redundant.

Use color to reinforce established meanings. Some colors denote meanings because of prior knowledge. In some societies, for example, red, yellow, and green have very definite meanings, respectively, as stop or alert, caution, and go or satisfactory conditions. To maximize the transfer of knowledge for both new and experienced users, you should be aware of such established meanings and include those that apply to your design.

The color that CUA assigns to an interface element might conflict with the standards of the country in which an application will be used. In that case, you may change the color assignment to one that is acceptable. However, be sure that the color you assign does not conflict with other CUA color assignments and that you use the substitute color consistently. For example, if the use of red, as defined by CUA, is not acceptable, you may assign a different color to the CUA element for which CUA assigns red.

Allow users to tailor colors. Selection of colors can be very subjective, and color preference might change eventually. This requirement to allow users to change colors is an example of the user interface principle of putting users in control.

The recommended uses of color for your CUA application can be found in Appendix B, "Color and Emphasis Table" in topic APPENDIX1.2.

2.5.3 Interaction

Interaction is the means through which users interact with the user interface components. Interaction is fundamental to establishing and reinforcing the users' conceptual model.

Subtopics

2.5.3.1 Object and Action Selection

The distinction between *objects* and *actions* not only affects the presentation of information but also how users interact with these different user interface components. The difference between the selection of objects and actions is significant.

Object Selection

Selection when applied to one or more objects should be the simple act of identifying objects. What is important is that users should always see a visual indication that selection occurred.

Selecting an object should never imply an action, affect the object itself, or commit users to anything. De-selecting an object should be as simple as selecting the object. Users should be able to cancel a wrong selection or change their minds without penalty. This is characteristic of a forgiving user interface.

Users can select an object by typing a selection character into the choice entry field immediately preceding it. If users then press the Actions function key, the cursor is placed into the action bar, but no action is performed on the selected object.

Action Selection

Selection when applied to an action should cause something to occur immediately and should also provide a visual cue. For example, if users want to save a document (perform the Save *action* on the *object*, a document), they move the cursor to the action bar choice File and press the Enter key to display the File pull-down. Users type the number of the Save choice into the choice entry field and press the Enter key. The document is saved immediately and a visual indication is given that the action occurred.

If users attempt to select an action before they select an object, the application presents a message that prompts users to select an object and then an action.

The difference between selecting objects and actions is that selecting an object causes nothing to happen except visual confirmation of the selection, whereas selecting an action causes the action to occur immediately and also provides visual confirmation that the action occurred.

Applications may allow an object to remain selected even after an action has been performed on it. If you expect users to perform another action, or a series of actions, on that object, then it should remain selected. If, however, as the result of an action, the object no longer exists in its original form and location, the object is no longer shown as selected.

2.5.3.2 Selection Techniques

There are two types of selection: *explicit* and *implicit*.

In *explicit selection*, users select a choice by typing a selection character in the choice entry field for the desired choice. For example, to explicitly select a choice in a single-choice selection field, users type the number of the desired choice into the choice entry field. For detailed descriptions of explicit selection techniques, see "Selection Elements" in topic 3.3.2.

In *implicit selection*, users simply move the cursor to the desired choice. The choice is automatically selected. When the Enter key is pressed, the application processes the panel.

Implicit selection is the technique supported in the action bar.

2.5.3.3 Selection Indicators and Emphasis

As users interact with an application, they need to know which choices are currently selected, which choices are not available for selection, the current state of choices (whether the choices are active or not), and which choices are associated with an error condition. CUA, therefore, provides the following visual cues for users:

Selection indicators and selected emphasis
Unavailable emphasis
Error emphasis.

Selection Indicators and Selected Emphasis: When users select a choice, the application should give some visual acknowledgement that the choice is selected. This feedback is provided by placing characters or symbols, referred to as *selection indicators*, into entry fields, or by using color and highlighting, referred to as *selected emphasis*. The visual acknowledgement can be different, depending on the type of selection element. For specific descriptions of selection indicators and the types of selected emphasis, see "Selection Elements" in topic 3.3.2.

Unavailable Emphasis: *Unavailable emphasis* is a visual cue that shows users that a choice is unavailable because some condition of the application does not allow them to select it. You must let users know when a choice is unavailable for selection. For a specific description of unavailable emphasis in various nonprogrammable terminal environments, see Appendix B, "Color and Emphasis Table" in topic APPENDIX1.2.

Do not confuse *unavailable* with *unauthorized*. Some applications provide several levels of function for different categories of users. Users in one category may not be authorized to use functions at a certain level. Choices that are connected to this *off-limits* function are considered unauthorized by CUA, not unavailable.

Error Emphasis: *Error emphasis* is a visual cue to users that they have incorrectly typed information into an entry field. Error emphasis consists of a change in the appearance of the incorrectly-typed data. The error emphasis characteristics for various nonprogrammable terminal environments, are described in Appendix B, "Color and Emphasis Table" in topic APPENDIX1.2.

2.5.3.4 Fast Paths

CUA defines several interaction techniques that can allow users to perform tasks more quickly and, therefore, more productively. These alternate techniques are referred to as *fast paths*.

You may assign application actions to function keys. These function keys are known as *accelerators*. By pressing these keys, users can initiate the associated actions immediately, without typing commands or accessing the action bar and pull-downs.

Typing commands in the command area can be a fast path for experienced users. They might be able to issue an application command faster by typing it than by selecting an object and then specifying the action (command name) and its parameters through the action bar, pull-downs, and associated pop-ups.

2.5.4 Action Bar Common Actions

Consistency of similar actions is important in reinforcing the users' conceptual model. For example, even though text editors and electronic mail applications are quite different, they both involve creating and editing objects. Therefore, both text editors and electronic mail applications must provide users with the ability to save and retrieve objects. CUA has defined common actions that you may apply to different kinds of objects. These actions are provided through the action bar and the function key area. All common actions, however, may not be applied to all types of objects, so CUA gives you guidelines for deciding when to use them. Action bar common actions are implemented through standard action bar pull-downs and dialog pop-ups.

Subtopics
2.5.4.1 Standard Action Bar Pull-Downs
2.5.4.2 Dialog Pop-Ups for File Pull-Down Choices

2.5.4.1 Standard Action Bar Pull-Downs

CUA has defined common actions and associated pull-downs for File, Edit, and Help. CUA also provides guidance for View and Options. The common actions usually apply to the object in the work area of a panel. This section provides guidelines on when to include these actions in your application. If your application supports File and Edit, they should be the first two choices in the action bar. If you include Help in your application, it is the last choice in the action bar. Figure 16 shows an action bar with the standard choices.

PICTURE 16

Figure 16. Standard Action Bar Choices. The common actions supported by your application are displayed as choices in the action bar pull-downs.

CUA defines a standard layout for the File, Edit, and Help pull-downs. This *layout* includes the text for the pull-down choices. Standard accelerators are also provided for some of the more frequently used functions. *Accelerators* are function keys that invoke the associated action without accessing the action bar and a pull-down. You should also use accelerators for application-specific actions that are used frequently.

Following is a description of the three standard pull-downs, File, Edit, and Help, and a general description of the View and Options actions. Each description includes general information and a figure showing the pull-down layout. For File and Edit, each figure is followed by a description of how your application must implement each pull-down action.

Subtopics
2.5.4.1.1 File Pull-Down
2.5.4.1.2 Edit Pull-Down
2.5.4.1.3 View Pull-Down
2.5.4.1.4 Options Pull-Down
2.5.4.1.5 Help Pull-Down

2.5.4.1.1 File Pull-Down

The *File* pull-down enables users to manipulate stored objects, such as files, as a whole. Every application that manipulates stored objects must provide the File pull-down. Some actions in the File pull-down, such as *New*, *Open*, and *Exit*, have the potential to lose unsaved changes. Applications, therefore, must be able to recognize this situation and display a message telling users that their unsaved changes will be lost. The application should also ask users if they want their changes saved. If so, the application should perform a save, as if users had selected the *Save* action from the File pull-down.

Figure 17 shows the actions that must be included in the File pull-down. Text that follows the figure describes each of these actions.

PICTURE 17

Figure 17. File Pull-Down

The File actions are organized by task: selecting actions, saving actions, and output actions. You may add to this pull-down other actions that involve manipulating an object. For example, if your application provides a Plot action, you would include the Plot action in the File pull-down, because plotting involves manipulation of an entire object. You should group *Plot* with the *Print* choice because Plot writes the object to an output device, a plotter. Some other application-specific actions that may be included in the File pull-down are Move, Copy, and Discard.

Following are descriptions of the File pull-down actions:

New allows users to create a new object. Existing information is removed from the work area, the current object name in the panel title is replaced, and the application object is displayed in the work area. The current object name in the panel title is replaced with *Untitled* or an application-generated object name that is meaningful to users. The application should ensure that an application-generated object name does not already exist.

Open reads a stored object and displays it for users to manipulate. If an object is not selected already in the work area, an application must prompt users for the name of the object to open, using a dialog pop-up. The Open action is one situation in the text subset where an action-object approach is allowed.

Save writes the existing object to a storage device, such as a disk. When an object is untitled, as it might be after requesting the *New* action, an application must prompt users for the object name to be used for the save, using a dialog pop-up. The work area of the panel remains unchanged. The current state of the application options should be saved with the object. In this way, when users reopen the object, they will find the environment exactly as it was when they saved the object.

Save as writes the existing object into a new object without changing the original one. An application prompts users for a new object name, using a dialog pop-up. The application then writes the new object to a storage device, using the new object name. The Save as action assumes users want to save to a new object. Therefore, an application must display a warning message that data is about to be lost if users indicate an existing object name as the new object name. The current state of the application options should be saved with the object, as it should be saved when the Save action is completed. After the object is saved, the panel title is updated to reflect the new object name.

Print prepares an object for printing and schedules it to be printed on a user-specified printer. You may display a dialog pop-up to request more information if your application supports specific print options.

Exit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. If you decide the File pull-down is not needed by your application, the Exit action must appear as the last choice in your first pull-down. This provides positional consistency for users because they will always find Exit in the same place.

2.5.4.1.2 Edit Pull-Down

The *Edit* pull-down allows users to perform electronically the types of actions that are done to reorganize *paper documents* in the real world. Paper documents can be organized in different ways by cutting, copying, and pasting sections of a document to create a new document. For example, when you want to move a paragraph, you cut it out of one section and paste it into another section. When you want to copy a paragraph from one part of a document to another, you make a copy of that paragraph, using a copy machine, and paste it into another location. When you want to delete something, you simply cut it out and throw it away. The *Cut*, *Copy*, and *Paste* actions, therefore, relate to real-world activities.

This real-world metaphor for paper documents is transferred to the computer environment for electronic documents through the actions available in the Edit pull-down. For example, users select a paragraph they want to move and then select the Cut action in the Edit pull-down. The selected paragraph disappears from its location in the document. Users select a destination point in the document and select the Paste action in the Edit pull-down. The paragraph is inserted at the new location.

The Edit pull-down contains common editing actions that apply to many types of objects. Any application that supports editing must provide these common editing actions in the Edit pull-down.

Internally, applications execute these actions by copying data to and from a *clipboard* in a standard data format. In the nonprogrammable terminal environment, the term clipboard refers simply to the buffer the system uses for temporary storage of data that is being manipulated. It does not necessarily imply a system clipboard function of the type supported by OS/2* and described in *Systems Application Architecture, Common User Access: Advanced Interface Design Guide*, SC26-4582.

You may add application-specific actions to the Edit pull-down, as appropriate. Figure 18 shows the common actions that are included in the Edit pull-down. Following the figure is text describing each of these actions.

PICTURE 18

Figure 18. Edit Pull-Down

Following are descriptions of the Edit pull-down actions:

Undo reverses the most recently executed user action. Because the Undo action deals with hidden objects, it should be modified *dynamically* to reflect exactly what is being *undone*. For example, when the last action executed by users was *Cut*, and users select the Edit pull-down, users should see *Undo cut* as the first pull-down choice.

For applications that manipulate text objects, such as documents, the following Undo actions, at a minimum, should be supported:

- *Undo paste* removes the text that was pasted and formats the document so it appears as it was prior to the paste. The selection state of the text should also be restored. Paste data remains on the clipboard.

- *Undo cut* re-inserts the text that was cut, formats the text, selects the text, and removes the text from the clipboard.

- *Undo clear* (if Clear is supported) re-inserts the text that was cleared, formats the text, and selects the text.

- *Undo typing changes* restores selected text after users have selected a block of text and typed new text to replace the selected text.

For applications that manipulate other types of objects, the application should customize the Undo actions as appropriate for the object type.

Mark selects, or marks, the portion of the object to be processed by a subsequent Cut, Copy, Paste, Clear, or Delete operation. If no portion of the object is currently selected, Mark selects and

emphasizes the position in the work area where the cursor is located when users switch to the action bar or press the function key assigned to Mark. If a portion of the object is already selected, Mark selects and emphasizes all positions between the previously selected portion and the current cursor position, inclusive. The selected portion of the object is emphasized using selected emphasis, as described in Appendix B, "Color and Emphasis Table" in topic APPENDIX1.2. To use the Mark pull-down choice, users must switch to the action bar using the Actions key, because that is the only way the application can know where the cursor was located when the switch to the action bar occurred.

Cut copies the selection to the clipboard and removes it from the object being edited. Depending on the type of object being edited, the space occupied by the removed portion may be either retained or compressed. For example, text applications usually compress the space, whereas graphics drawing applications usually leave the space blank.

Copy produces a duplicate of the selected portion of an object on the clipboard without removing the selected portion from the object that is being edited.

Paste copies the contents of the clipboard into the object being edited at the location selected. For text applications that support word wrap, information to the right and below the selected location is shifted and formatted. However, for certain applications, such as graphics drawing applications, it may be appropriate to overlay the information.

Unmark removes the emphasis from the currently selected portion of the object, de-selecting it.

Clear (optional) removes the selected portion from the object without copying it to the clipboard. The space is not compressed.

Delete (optional) removes the selected portion from the object without copying it to the clipboard. The space is compressed.

2.5.4.1.3 View Pull-Down

The *View* pull-down allows users to select different ways to look at an object without affecting the object itself. This pull-down contains actions that control such viewing specifications as how much information is presented, the presentation order, the format, and the scale. For example, Figure 19 shows an example of a View pull-down as a way for users to view the list of items in the in basket of a mail application.

PICTURE 19

Figure 19. View Pull-Down

In Figure 19, **All** and **Some...** would be mutually exclusive, providing users the capability of viewing the entire list or only selected items. The selection criteria for the Some view is determined through a dialog pop-up that is provided when users select the Some action. The last set of choices allows users to determine the order in which displayed items appear. If users select the **By other...** choice in the View pull-down and press the Enter key, sort criteria other than subject, date, or name will be presented in a dialog pop-up. The **1** in the choice entry field that precedes **All** tells users the current state of the view of the object; it tells them that all of the items in the in basket are displayed. The **2** in the choice entry field that precedes **By subject** tells users the current state of the display order of the items; it tells them that the items in the in basket are displayed in order by date.

A calendar application provides a very different example of a View pull-down because such an application may allow users to select viewing their calendars by day or by month.

You can see by the previous examples that the content of the View pull-down is specific to the type of object being handled by the application. Therefore, the View pull-down cannot be standardized. CUA requires that you provide the View pull-down in your application if you need to give users the ability to view their objects in different ways. Your application should save the current state of the view with the object.

2.5.4.1.4 Options Pull-Down

Provide the *Options* pull-down if users need a way to customize the object itself. Like the View pull-down, the Options pull-down is specific to a particular application, so its contents cannot be standardized.

For example, in a document editor, you may allow users to display documents in memo, letter or report style. In Figure 20, the **2** in the choice entry field indicates that users previously chose to display documents in the **Letter** style. This style remains in effect until users select a different choice. Use of the Options pull-down can give users more control of the environment.

PICTURE 20

Figure 20. Options Pull-Down

2.5.4.1.5 Help Pull-Down

The Help pull-down provides users with access to various kinds of help information. All applications that access online help information must provide the Help pull-down. Figure 21 shows the actions that are included in the Help pull-down of panels in the primary window of an application.

PICTURE 21

Figure 21. Help Pull-Down for Panels in Primary Windows

For details about the implementation of help, including descriptions of the Help pull-down actions, see Chapter 15, "Help" in topic 3.10.

2.5.4.2 Dialog Pop-Ups for File Pull-Down Choices

The Open and Save as actions described in "File Pull-Down" in topic 2.5.4.1.1 each require a dialog pop-up for requesting further information from users.

Subtopics
2.5.4.2.1 Open
2.5.4.2.2 Save As

2.5.4.2.1 Open

An *Open* dialog pop-up is displayed when users select the Open... choice from the File pull-down. An application uses this dialog pop-up to determine which object to open and display for users. The Open dialog pop-up also applies to applications that have a relatively small number of selectable objects for which a single search criteria may be used to locate the objects. Figure 22 shows an example of an Open dialog pop-up. This example should be tailored to address the specific requirements of your application or environment.

PICTURE 22

Figure 22. Example of an Open Dialog Pop-Up

In Figure 22, at the top of the work area in this Open dialog pop-up, there is an entry field into which users can type the name of the object they want to open. The selection list following this entry field is a list of objects from which users can select one object to open. Users select an object to open either by typing the name of the desired object into the entry field or by typing a selection character over the period that precedes the desired object name in the selection list.

Pressing the Enter key causes an Open action to be attempted. If users have not selected an object, or if there is a conflict between the object name in the entry field and the object selected in the list, the application presents a message informing users of the condition and telling them what corrective action is needed.

The Open dialog pop-up may contain additional, application-specific options related to the Open action.

Users can select Cancel or Help at any time during the dialog.

2.5.4.2.2 Save As

A *Save As* dialog pop-up is displayed when users select the Save as... choice in the File pull-down. This dialog pop-up is also used for the Save action if the object is not named. Your application uses this dialog pop-up to determine the object name users want the application to use when the application stores the object that is currently in the work area. Figure 23 shows an example of a Save as dialog pop-up. This example should be tailored to address the specific requirements of your application or environment.

PICTURE 23

Figure 23. Example of a Save As Dialog Pop-Up

In Figure 23, at the top of the work area, there is an entry field into which users type the name under which an object is to be stored. A selection list follows this entry field, allowing users to select a folder to contain a saved object. Users select a folder by typing a selection character over the period that precedes the object name in the selection list.

The Save as dialog pop-up may contain additional, application-specific options related to the Save action. For example, options could be added to allow users to choose *Read only* or *Read/Write* attributes for the object being saved.

Pressing the Enter key causes a Save action to be attempted.

Users can select Cancel or Help at any time during the dialog.

2.5.5 A Sample Text Subset Application

Here is a simple text subset application scenario. It is from a hypothetical application that provides a department manager with four kinds of data and the ability to manipulate the data in several ways. The scenario illustrates implementations of many CUA design principles and interface components. Various interaction techniques are used in the scenario for illustrative purposes.

This is a *very* simple example, intended primarily to convey the general flavor of the text subset. The scenario shows only one of many possible design approaches for such an application. Depending on the needs of the enterprise and the types of users the application would serve, other designs might be appropriate.

The first screen of the application is shown in Figure 24. It contains the name and copyright statement of the application. The copyright statement appears in the message area. It also contains a *single-choice selection field* with choices that identify the kinds of objects that the manager can work with.

PICTURE 24

Figure 24. Initial Screen of The Electronic Manager Application

The manager types a **3** in the *choice entry field* to select **Personnel data** and presses the Enter key. A **Personnel data** panel appears in the primary window. Because a specific personnel data object has not been selected and opened, the work area of this panel is empty.

The manager switches the cursor to the *action bar* by pressing the F10 key and then presses the Enter key. The File pull-down is displayed, as shown in Figure 25. The manager selects ***Open...*** by typing a **2** in the pull-down choice entry field.

PICTURE 25

Figure 25. File Pull-Down. ***Open...*** is selected. ***New***, ***Save***, and ***Save as...*** are currently unavailable. The manager knows these choices are unavailable because asterisks appear in place of their choice numbers. Note that the application assigns the F11 key as the ***accelerator*** for ***Print***.

The manager presses the Enter key and the Open *dialog pop-up*, containing a *single-choice selection list*, appears, as shown in Figure 26. The manager selects employee **Harriet R. Easton** by positioning the cursor on the period that precedes her name and typing a slash over the period.

PICTURE 26

Figure 26. Open Dialog Pop-Up

When the manager presses the Enter key, the employee personal data panel for Harriet Easton is displayed, as shown in Figure 27. The panel contains a mixture of *entry fields* and *protected text*. The manager could change the information in the entry fields, such as the employee address, but could not change, for example, the last performance rating.

PICTURE 27

Figure 27. Employee Personal Data Panel

The manager wants to see a different portion of the employee's data record. After the manager switches the cursor to the action bar, moves the cursor to the View action bar choice, and presses the Enter key, the View pull-down appears, as shown in Figure 28. There is a **1** in the pull-down choice entry field, indicating that the current view of the data is the **Personal data** view.

PICTURE 28

Figure 28. View Pull-Down

The manager types a **3** in the choice entry field to select the **Awards** view, and presses the Enter key. The current view is now Awards (3). When the pull-down is displayed the next time, a *3* will appear in the choice entry field.

As shown in Figure 29, the data about Harriet Easton's awards appears in place of her personal data.

PICTURE 29

Figure 29. Employee Awards Data

To recommend an award for Harriet Easton, the manager moves the cursor to the **Manage** action bar choice and presses the Enter key. The Manage pull-down appears, as shown in Figure 30. That figure shows that the manager has typed a **2** in the pull-down choice entry field to select **Recommend award...**. The ellipsis (...) following **Recommend award** indicates that a dialog pop-up will result from the selection of this choice.

PICTURE 30

Figure 30. The Manage Pull-Down. ***Recommend award...*** is selected.

When the manager presses the Enter key, a pop-up appears, as shown in Figure 31. From this pop-up, the manager can select an award type. To recommend a leadership award, the manager types a **3** in the choice entry field.

PICTURE 31

Figure 31. Dialog Pop-Up Resulting from the Manage Pull-Down

The manager presses the Enter key and the **Employee Leadership Award Recommendation** form appears in another pop-up, as shown in Figure 32. The manager fills in the first page of the form.

PICTURE 32

Figure 32. First Page of Employee Leadership Award Recommendation Panel

When the manager presses the F8 key, the panel scrolls to display the rest of the information as shown in Figure 33.

PICTURE 33

Figure 33. Second Page of Employee Leadership Award Recommendation Panel

The manager wants additional information about how to select award supplies. With the cursor on the **Award supplies** selection field, the manager presses the F1 key. The help pop-up shown in Figure 34 appears.

PICTURE 34

Figure 34. Help Pop-Up for Award Supplies

The manager moves the cursor out of the pop-up and selects award supplies by typing a slash character (/) in the entry fields that precede the desired supplies. Selection of the award supplies completes the form. When the manager presses the Enter key, the help and recommendation form pop-ups disappear, and the application submits the recommendation for processing. The employee awards data is re-displayed, as in Figure 29. The manager exits the Personnel data function by pressing the F3 key. The

initial screen of The Electronic Manager is re-displayed.

2.6 Chapter 4. Entry Model

Although the text subset of the graphical model is generally preferred for all new applications on the nonprogrammable terminal, there are some applications whose structure and objectives may warrant an exception to that rule. They are:

An application with only limited actions that can be performed on the primary object or objects in the application. This is true of many data-entry-intensive applications where the only application action available is *Enter data*. This type of application does not need an action bar because only a limited number of actions are available to users. Either the actions or the objects in this type of application may be implicit, or the actions and objects are combined. These limited-action applications typically are designed to maximize the resources of a host system by minimizing the required number of host interrupts.

A new release of an existing action-object-oriented application that does not involve a major re-design of the application. One of the prime objectives for this style of application might be to retain consistency with previous versions for users who will be upgrading to a new version. However, components being used in the application, such as *entry fields* and *selection fields*, should be revised to conform to CUA. These applications also might have been designed to minimize the number of host interrupts generated by the application.

An application that should be action-object, as determined by a thorough task analysis. This decision may have been made because the intended users have a current conceptual model of the application that is action-object.

An application that is designed to be *walk-up-and-use* by users who are constantly new, and, therefore, will not be trained. Such applications must control the users' choices very closely. An example is an information display in a hotel or airport lobby. These applications typically combine object and action choices. They might be object-action or action-object, but they would not include an action bar and, therefore, would be entry model applications.

The intent in describing an entry model is to define a level of CUA conformance for applications that, for the reasons listed above, cannot be designed as text subset applications. By following the entry model, these applications might gain the advantage of user interface consistency and provide an easier migration for users who also may use applications with other CUA interface styles.

Subtopics

2.6.1 Differences between Entry Model and Text Subset

There are two primary differences between the text subset and the entry model:

- Entry model applications are not required to be object-action oriented.
- Entry model applications do not have an action bar.

The key difference for users between a text subset application and *some* entry model applications will be in the process sequence: action-object instead of object-action. The process sequence makes a significant difference in the structure of an application. It might mean that users are presented with a hierarchical menu-driven application. It also might mean that users choose actions first and then objects, or actions and objects that are combined. The distinction between action selection and object selection that was made in the text subset model is, therefore, no longer valid.

In terms of presentation and interaction, the entry model includes most of the components of the text subset, except for the action bar. The other components that were illustrated in the text subset look and act the same way in the entry model. However, the work area of an entry model panel may contain both objects and actions.

The use of pop-up windows for dialogs, prompt lists, messages, and help is optional in the entry model. Dialogs that would appear in a pop-up in a text subset application may occupy the full screen in an entry model application.

The following example of an entry model application shows a simple data-entry application, and points out the basic process sequence, interaction, presentation, and components found in entry model applications. A more detailed description of the rules governing the presentation and interaction techniques of the components described here is found in Part 3, "Component Descriptions" in topic 3.0.

Like the text subset sample application, this example of an entry model application is a very simple example, which is primarily intended to convey the general flavor of the entry model. The scenario only shows one of many possible design approaches for this type of application. Depending on the needs of the enterprise and the types of users the application would serve, other designs might be appropriate.

Note: This same application is used in the next chapter to show how an application would look after it has been migrated to the programmable workstation.

2.6.2 Entry Model Example: A Simple Data-Entry Application

The following simple application contains some examples of entry model panels, using components described in this book. This type of application might be used to maintain records in a doctor's office. The initial panel is the main menu for the application. It presents users a list of items they can choose from. This list is an example of a single-choice selection field. In this example, the objects are forms. Each form has its own implied action.

PICTURE 35

Figure 35. Single-Choice Selection Field

In Figure 35, the choices are numbered and the first choice is preceded by a *choice entry field*, designated by an underscore, so users can type the number of their choice. Note that, in this example, the first choice is the default choice, so its corresponding number (1) is in the choice entry field. Pressing the Enter key without typing a number processes the designated default choice. Below the work area is a message area, followed by the command area and the function key area.

If users select choice 1 from the panel in Figure 35 and press the Enter key, they will see the panel in Figure 36.

PICTURE 36

Figure 36. Data-Entry Form. This form contains entry fields and a selection field.

In Figure 36, the data-entry form in this panel is a collection of entry fields and one selection field. Note the standard locations for the panel title, scrolling information, message area, command area, and function key area. **Marital status** is a single-choice selection field; all the other fields are entry fields.

The *More* designator tells users that a portion of the panel in Figure 36 is scrollable. The F7 and F8 keys are displayed in the function key area.

Assume that users are entering information about a new patient into the form in Figure 36. Users have pressed the F8 key to scroll to the additional information and have completed all the required information, except for the Insurance ID number. If users then press the Enter key from the last **New Patient Information** panel, they see the panel shown in Figure 37.

PICTURE 37

Figure 37. Action Message in Message Area

The panel containing the **ID number** entry field has been re-displayed with a message in the message area that tells users there is an error, as shown in Figure 37. The cursor is positioned in the field containing the error, and that field is displayed with error emphasis. The message is displayed in the message area instead of a pop-up because the message itself does not require a response.

If users then press the F1 key while the cursor is in the ID number field, users see the panel shown in Figure 38.

PICTURE 38

Figure 38. Help Panel

Help panels may be displayed either as a full-screen panel or in a pop-up. Help is provided on the field where the cursor was positioned when Help was requested. From the help panel itself, users can request Extended help (Ex help), Help index (Index), and Keys help, as shown in the function key area of Figure 38. F12 cancels help, and users are returned to the panel from which they requested Help, that is, to Figure 37. After users complete all the information in Figure 37 and press the Enter key, the users are returned to the main menu in Figure 35.

If users select choice 4 from the main menu in Figure 35, they see the panel in Figure 39.

PICTURE 39

Figure 39. Prompt for Entry Field

In Figure 39, the plus sign (+) to the right of the **Patient name** field indicates that there is a prompt list available for this field, through the Prompt function. To see the list of names for selection, users press the Prompt key (F4) when the cursor is in the **Patient name** field. Users see the list in Figure 40.

PICTURE 40

Figure 40. Prompt List Pop-Up

Because this is an entry model application, the prompt list panel could occupy the full screen. In this example, however, this panel is presented in a pop-up. The prompt list in Figure 40 is an example of a single-choice selection list. To scroll the list, users press the Backward or Forward keys (F7 or F8). In this type of field, users select items by typing a slash character (/) over the period to the left of the item they want to select. When users press the Enter key, the selected name is placed automatically in the appropriate entry field, as shown in Figure 41. The prompt list pop-up disappears and users are returned to the entry field from which they requested Prompt.

PICTURE 41

Figure 41. Multiple-Choice Selection Field

Figure 41 contains an example of a multiple-choice selection field. In this type of field, users select items by typing a slash character (/) in the entry field to the left of the item they want to select. Users can select more than one item in a multiple-choice selection field. After all selections have been made, users press the Enter key. If users select **Immunizations** from the selection field in Figure 41, they see the panel in Figure 42.

PICTURE 42

Figure 42. Multiple-Choice Selection List

Figure 42 contains an example of a *multiple-choice selection list*. This list is similar in appearance to a multiple-choice selection field, except that it is scrollable and the data contained in it may be variable. Users can move from choice to choice with the arrow keys or the Tab key. Selections are made by entering the slash character (/) in the entry field preceding each choice. When all selections have been made, users press the Enter key and the panel is processed.

2.7 Chapter 5. Migration to the Programmable Workstation

This chapter describes differences that will occur in the user interface of a text subset or entry model application that is migrated from the nonprogrammable terminal environment to the programmable workstation environment *without* re-designing the user interface. The descriptions assume that the application will run under OS/2* Version 1.2 or greater on the programmable workstation.

Applications that are being designed specifically for the programmable workstation should use an object-action design approach and follow the rules and guidelines for the graphical model in *Systems Application Architecture, Common User Access: Advanced Interface Design Guide*, SC26-4582.

To describe the migration, some of the sample screens that were used to describe the entry model and the text subset are repeated, with the differences pointed out, so you can easily compare the two screens. For detailed information about any of the OS/2* components, see the *Advanced Interface Design Guide*.

Finally, there is a summary of the differences, presented in table format at the end of this chapter.

Subtopics

2.7.1 Programmable Workstation Differences

In the programmable workstation, or OS/2*, environment, the primary visual components of the user interface are screen background, windows, icons, and a free-moving mouse pointer. The appearance and behavior of these components is enforced automatically by OS/2* and the appearance is much more graphical than on the nonprogrammable terminal. Within windows, standard interface components, such as selection fields, contain graphical elements, such as check boxes and radio buttons, which are made possible by the graphical environment of OS/2*.

Some other differences are:

- The size and location of the primary window that is used to display the application panels can be controlled by users and will not occupy the full screen. If users reduce the size of the application window, scrollable areas are reduced in size first, then the content of the window may be clipped. When users reduce the size of a window using the keyboard, clipping of the information in the window begins at the bottom and the right portions of the window.

- Users can communicate with the computer through the mouse as well as the keyboard. CUA defines interaction techniques for both devices. Because either the keyboard or a mouse may be more efficient in a specific situation, users can switch between the keyboard and a mouse almost any time in a dialog without having to change application modes. For detailed information on mouse interaction techniques, see the *Advanced Interface Design Guide*.

- Text displayed on the screen may be proportionally spaced on the programmable workstation.

- The programmable workstation supports implicit selection in single-choice selection fields and lists in the work area, as well as in in the action bar.

- The programmable workstation does not support multiple single-choice selection fields in pull-downs. The same effect can be achieved by users with serial single selections.

- Dialog pop-ups are referred to as *dialog boxes* and are movable.

- On the programmable workstation, there are 5 types of emphasis. In addition to the four previously discussed, there is also cursored emphasis, which can be combined with the other four types.

- Entry fields appear as boxes instead of underscores.

- Radio buttons and check boxes replace the choice entry field in selection fields as visual indicators of the currently selected choice.

- Pushbuttons replace the function key area on the programmable workstation. Pushbuttons are an indication to users that the actions in the pushbuttons can be selected using the mouse.

- The programmable workstation supports mnemonic selection in place of numeric selection in selection fields and in the action bar and its pull-downs. A *mnemonic* is a single character that provides a fast interaction technique for selecting choices from the keyboard. When users type a valid mnemonic, the selection cursor moves to the choice that the mnemonic is assigned to and the choice is automatically selected or de-selected, as appropriate. Users can type alphabetic mnemonics in either uppercase or lowercase. A mnemonic is underlined to provide a visual cue to users that mnemonic selection is available.

These and other differences are pointed out in the sample application screens that follow.

2.7.2 Sample Programmable Workstation Screens

If the sample entry model application described in Chapter 4, " Entry Model" in topic 2.6 is migrated to the programmable workstation, the initial panel in Figure 35 in topic 2.6.2 would look like the window in Figure 43.

PICTURE 43

Figure 43. Single-Choice Selection Field on Programmable Workstation

The panel appears in a movable, sizable window. The title bar of the window consists of the System Menu icon, the window title, and the window-sizing icons. Users can select the System Menu icon to display a pull-down containing actions users can perform on a window. The actions are: Move, Size, Minimize, Maximize, Restore, Close and Switch to. The panel title is displayed as the window title. Window-sizing icons provide a fast path to three of the system menu actions: Minimize, Maximize and Restore.

If users reduce the vertical size of the panel so that some of the choices are no longer visible, a scroll bar appears on the right side of the work area. (The *work area* is similar to the OS/2* *client area*.) If the panel is reduced in size horizontally so that the choices are no longer completely visible, a horizontal scroll bar appears.

The appearance of the single-choice selection field is changed: there is no entry field preceding the first choice and the choices are not numbered. Instead, *radio buttons* precede each choice to indicate the current selection; when an item is selected, the button is partially filled in. Each choice has an application-defined mnemonic that is underlined. The selection cursor on the programmable workstation is a dotted outline box that completely covers the choice. In a single-choice selection field, the cursored choice is shown with cursored emphasis (surrounded by a dotted outline box). For a detailed description of emphasis on the programmable workstation, see the *Advanced Interface Design Guide*.

Users can select an item by typing the mnemonic for that item, by moving the cursor to the choice (implicit selection), or by moving the mouse pointer to a selection and clicking mouse button 1.

The command entry field is displayed as a box rather than as underscores. All entry fields on the programmable workstation are displayed as boxes.

Notice that the function keys are displayed as pushbuttons on the programmable workstation. This is an indication to users that the functions in the pushbuttons can be selected by the mouse.

PICTURE 44

Figure 44. Scrollable Data Entry Panel

Figure 44 illustrates how the data entry panel shown in Figure 36 in topic 2.6.2 would look on the programmable workstation. This panel would be displayed if users selected **New patient information** from the main menu of Figure 43. All the entry fields are displayed as boxes. Entry fields are scrollable, except for those that contain fixed-length data, such as phone numbers.

The text cursor indicates the keyboard input location for text entry. On the nonprogrammable terminal, the text cursor is an underscore; on the programmable workstation, the text cursor is a thin vertical bar. The default input mode is insert mode on the programmable workstation. (The default is replace mode on the nonprogrammable terminal). Cursor movement on the programmable workstation is restricted to valid fields. In Figure 44, if users press the Down arrow key, the cursor moves to the next valid field below the current field. For example, if the cursor is in the **Zip code** field and users press the Down arrow key, the cursor moves to the **Birth date** field, as if users had pressed the Tab key. If users press the Up arrow key from the **Zip code** field, the cursor moves to the previous valid field, **City, State**, as if users had pressed the Back Tab key.

In the single-choice selection field, **Marital status**, radio buttons precede the choices. If users press the Down arrow key in that field, the cursor moves to the next choice in the field. For example, if users press

the Down arrow key while the cursor is on **Married**, the cursor moves to **Single**. If, however, users press the Tab key, the cursor moves to the next field. For example, if users press the Tab key with the cursor on **Married**, the cursor moves to the **Home phone** field.

The scrolling information located at the top of the panel on the nonprogrammable terminal has been replaced with a scroll bar, to indicate that the area of the window between the separators is scrollable. (On nonprogrammable terminals, these separators are known as *panel area separators*.) Users can scroll the panel forward by using the mouse on the scroll bar, by pressing the Down arrow key or Tab key while the cursor is at the bottom of the panel, or by pressing the Page Down key.

PICTURE 45

Figure 45. Message Pop-Up

Figure 45 illustrates how a message would appear on the programmable workstation. Messages are displayed in pop-ups, rather than in the message area as it is on the nonprogrammable terminal. These pop-ups are referred to as *message boxes* in OS/2*. The pop-up is positioned so it does not cover the field in which the error occurred. Pushbuttons are located at the bottom of the pop-up. The cursor is positioned on the **OK** pushbutton. Users can press the Enter key or click on the OK pushbutton with the mouse to remove the message pop-up. The cursor is then returned to the field in which the error occurred. If users then request **Help**, the help panel is displayed as shown in Figure 46.

PICTURE 46

Figure 46. Help Panel

The help panel is displayed in a *help window* that is separately movable and sizable. Just as a message pop-up is carefully placed, the help window is positioned so it does not overlay the field for which help was requested. Because of the help facility used in OS/2*, the help window contains an action bar, rather than a function key area, for access to additional help functions.

Figure 47 shows an example of a multiple-choice selection field on the programmable workstation. Choices are preceded by *check boxes*; an **x** in the box acts as a selection indicator. Selection is made by pressing the Spacebar, by clicking the mouse button, or by typing the mnemonic for a choice. De-selection is done by clicking the mouse button or by pressing the Spacebar while the cursor is positioned over a choice.

PICTURE 47

Figure 47. Multiple-Choice Selection Field

If users request the Prompt function while the cursor is in the **Patient Name** field in Figure 47, they will see the window shown in Figure 48.

PICTURE 48

Figure 48. Prompt List Pop-Up

The prompt list in Figure 48 is a single-choice selection list. An OS/2* list box can be used to display selection lists. Users can move the pop-up that contains the list. The scroll bar indicates that the list is scrollable. Users select a choice by moving the cursor to the choice or by clicking on it with the mouse. When the pop-up is processed, the selected information is automatically entered in the panel.

Figure 49 shows an example of a multiple-choice selection list on the programmable workstation. Multiple-choice selection lists can be displayed using list boxes on the programmable workstation. They are similar in appearance and interaction to the single-choice selection list on the programmable workstation, except that users can select more than one choice. Instructions can make this distinction clear to users. Users move from choice to choice with the Down arrow key (not the Tab key) and

make their selections by pressing the Spacebar while the cursor is positioned over a choice, or by positioning the mouse pointer over a choice and clicking the mouse button. Selected items are displayed with selected emphasis. The cursor keys, the Page Up and Page Down keys, or the scroll bar (with a mouse) can be used to scroll the list.

PICTURE 49

Figure 49. Multiple-Choice Selection List

2.7.3 Action Bar Differences

PICTURE 50

Figure 50. Action Bar and Pull-Downs

Figure 50 is an example of how a text subset action bar and pull-down would look on the programmable workstation. The action bar pull-down does not contain entry fields and the choices are not numbered. The check marks that precede the choices **All** and **By subject** indicate the current state of the view of the object. If there is more than one single-choice selection field in a pull-down, selection is accomplished through a series of single selections, rather than by selecting one choice from each group. Users, therefore, must re-display the pull-down in order to select another choice. The mouse may be used to select action bar and pull-down choices.

2.7.4 Programming Considerations

Following are some additional considerations that you might want to take into account as you migrate your application to the programmable workstation.

1. Assigning mnemonics: In each window, a mnemonic must be unique within the action bar, within a pull-down, and within the client area. If possible, assign the first letter of a choice as its mnemonic. If the first letter is already assigned, choose another letter in the choice that seems reasonable for the context of your application. If all the letters in a choice are already assigned or the choice consists of DBCS characters, you may choose another letter or a keyboard character, such as the comma (,). When you use a letter or character that is not part of the choice, enclose the mnemonic in parentheses immediately following the choice. Choices that appear many times in an application should always be assigned the same mnemonic.

2. Avoid using the full width of the panel because there must be room for the scroll bar on the right side of the screen. Do not use more than 76 columns for a panel so that the scroll bar will be visible on the screen.

3. When you are re-coding an application for migration to the programmable workstation, you might want to update the function key area. Some updates in consideration of the programmable workstation keyboard can improve the usability of the application.

   The following changes might be appropriate, depending on the application:

   Remove Forward and Backward key assignments and change F12=Cancel to Esc=Cancel. (F12=Cancel must be supported even if it is not displayed.)

   You may need to add pushbuttons on the programmable workstation for keys that were not displayed on the nonprogrammable terminal because they were engraved on nonprogrammable terminal keyboards, such as Help on the AS/400* keyboard.

   You should insert a pushbutton in dialog pop-ups. This pushbutton should name the default action for the pop-up.

   If your application used F13 through F24 or Set 2 on the nonprogrammable terminal, you may use the shifted state of function keys available on the programmable workstation (Shift+F1 through Shift+F12).

2.7.5 Summary of Differences

The following table summarizes the differences between the nonprogrammable terminal and the programmable workstation environments.

Component	**Nonprogrammable Terminal**	**Programmable Workstation**
Presentation	Full screen panel, pop-up	Sizable window, pop-up
Scroll indicators	More: Scroll bars	
Selection cursor	Cursor	Dotted outline box
Function key format	F1=Help	F1=Help with pushbutton border
Help	Pop-up or full screen panel with function key area	Help window with action bar, no function key area
Pop-up	Fixed	Movable
Entry field prompt	Leader dots	Colon (optional)
Single-choice selection field	*Appearance*: choice entry field, numbered *Interaction*: Number in choice entry field	*Appearance*: radio buttons *Interaction*: mnemonic, point-and-select (keyboard or mouse)
Multiple-choice selection field	*Appearance*: choice entry field before each choice in vertical list; choice entry field is an underscore (_) *Interaction*: / or country-designated character in entry field	*Appearance*: check boxes *Interaction*: Spacebar select, mouse click, mnemonics
Single-choice selection list	*Appearance*: Choice entry field before each choice in a vertical list with adjacent scroll information; choice entry field is a period (.) *Interaction*: / or country-designated character in choice entry field	*Appearance*: list box with scroll bars *Interaction*: Point-and-select (keyboard or mouse)
Multiple-choice selection list	*Appearance*: choice entry field before each choice in vertical list with adjacent scroll information; choice entry field is an underscore (_) *Interaction*: / or country-designated character in entry field	*Appearance*: list box with scroll bars *Interaction*: Spacebar select, mouse click
Entry fields	Not scrollable Underscored	Scrollable Solid-line box

Messages	Message area, pop-up	Pop-up
Text entry	Default=replace	Default=insert
Input device	Keyboard	Keyboard, mouse
Cursor movement	Unrestricted	Restricted to fields
Tab	Moves cursor to next unprotected field	Moves cursor to next field
Pull-down	Multiple selections per appearance of pull-down	Single selection per appearance of pull-down

3.0 Part 3. Component Descriptions

Subtopics

3.1 Chapter 6. Introduction to Basic Interface Components

Previous chapters introduced the principles and models for designing effective Common User Access interfaces. The chapters in this part contain rules, recommendations, and options for presenting and interacting with the fundamental components on which those interfaces are based.

The application developer presents application information to users in the form of *panels*. A panel may occupy the entire screen or be presented in a bordered area of the screen called a *pop-up*. Each panel is composed of one or more individual elements and areas, each of which must follow specific CUA rules for layout and content. The general location of various panel elements and areas is shown in Figure 51.

PICTURE 51

Figure 51. Panel Elements and Areas

If you design your application based on the text subset of the graphical model, every panel in the primary window must have an *action bar*. The action bar contains a list of choices that are arranged horizontally across the top of the panel. Selection of one of the action bar choices results in a *pull-down* that contains actions or properties that can be applied to objects in the work area.

The purpose of each panel is indicated by its *panel title*, which appears immediately below the action bar separator line. If there is no action bar, the title appears on the top line of the panel. The panel also may be uniquely identified by a *panel ID* that appears on the left side of the line that contains the panel title.

Every panel you design has a *work area*. As its name implies, the work area is the primary area of the panel through which users interact with the application. It begins immediately below the panel title separator and occupies most of the panel. A typical work area may contain *selection fields* and *selection lists* through which the application presents choices to users, *entry fields* in which users provide information to the application, *action lists* that present users a list of objects on which they can perform a variety of actions, *protected text* that provides information that users cannot modify, or a combination of any of these. It also may contain other panel elements that guide users in interacting with the application, such as *instructions, headings, field prompts*, and *descriptive text*.

If there is more information than can be shown on the panel at one time, all or part of the work area may be scrollable. *Scrolling information* is provided to indicate that more information is available and to show the direction in which the panel area can be scrolled.

At the bottom of the panel, below the work area, are the *message area* and the *function key area*. Some panels also may have a *command area* between the message area and the function key area.

Messages that indicate the status of the users' work or a condition needing the users' attention may be written to the message area or, under some conditions, to a pop-up that overlays part of the panel. Because the message area is usually blank, it may serve as a separator line between the work area and the command area or function key area.

The *function key area* identifies actions that can be performed by specific keys. It contains the *common actions* that apply to a specific panel and also may contain application-defined actions. The common actions include scrolling actions, such as Backward and Forward; navigation actions, such as Exit and Cancel; and actions requesting assistance, such as Help and Prompt. Some of the actions may be invoked also by selecting choices in an action bar pull-down or by entering commands in the command area.

The result of invoking an action may be a change in the current panel, the presentation of a different panel, or the presentation of a pop-up that extends the dialog with the current panel.

In some figures in this section of the book, the function key area shows the use of 12 function keys; in other figures, the function key area shows the use of 24 function keys.

Subtopics

3.1.3 Component Requirements by Model

3.1.1 Double-Byte Character Set Considerations

Japanese, Korean, and Chinese languages are not based on Latin characters. The characters in these languages are represented using two bytes of information instead of the single byte used in representing Latin-based languages.

If you design applications for countries that use the double-byte character set (DBCS), you have to consider additional rules pertaining to cursor location, entry fields, text spacing, and special symbols and characters.

Single-byte characters are also used with DBCS characters, so the guidelines found in this publication also apply to applications that use the DBCS.

Some information in this book about how to capitalize words does not apply to DBCS characters. However, because Latin-based characters are used also in DBCS countries, you should capitalize those characters according to the rules specified in this book.

3.1.2 Summary of Interface Components

The following tables summarize the characteristics of the interface components and the CUA requirements for using these components in the entry model and text subset of the graphical model.

Subsequent chapters provide the details of these components and the methods of interacting with them. Layout and content characteristics not specified in these chapters are left up to the application developer.

CUA also specifies certain color and emphasis characteristics of the interface components presented on a panel. These requirements are summarized in Appendix B, "Color and Emphasis Table" in topic APPENDIX1.2.

Table 1. Summary of Interface Components

Component	**Description**
Action bar	Lists choices that represent groups of actions that users request. Actions appear in pull-downs. It is used on all panels in the primary window of applications that are based on the text subset. See Chapter 10, "Action Bar and Pull-Downs" in topic 3.5.
Action bar pull-down	Extends from the action bar and displays groups of actions that users can select. It appears when users select an action bar choice. See Chapter 10, "Action Bar and Pull-Downs" in topic 3.5.
Action code	A number or alphabetic character entered into an action entry field to specify the action to be performed on the items in the action list. See "Action Lists" in topic 3.3.3.
Action entry field	An entry field that is associated with each choice in an action list. Users type a number, an alphabetic character, or a command into the action entry field to indicate the actions they want to apply to choices in the action list. See "Action Lists" in topic 3.3.3.
Action list	Provides a list of items that users can perform one or more actions on. Users specify actions by typing an action code or command in an entry field next to each selected item. See "Action Lists" in topic 3.3.3.
Choice entry field	An entry field used in selection fields and lists to select a choice. Users select a choice by typing a number or a slash character (/) into the choice entry field.
Command area	Provides space in which users type either application or system commands. It is indicated by the command prompt, **Command ===>**. See Chapter 11, "Command Area" in topic 3.6.
Copyright information	Displays information about the application copyright and ownership. See Chapter 17, "Copyright Information" in topic 3.12.
Cursor	A visual cue that shows users the current position of the keyboard input focus.
Descriptive text	Information located to the right of an entry field. It tells users about the kind of information that is required to complete an entry field.
Entry field	Provides space in which users type information. It is usually indicated by an underscore. See " Entry Fields" in topic 3.3.1.
Field prompt	Identifies entry fields, selection fields, and variable output information. See "Field Prompts" in topic 3.2.6.
Function key	Lists available actions assigned to keys. See

area	Chapter 12, "Function Key Area" in topic 3.7.
Heading	Identifies entry fields, selection fields, selection lists, action lists, and protected text. There are two types: column headings and group headings. See "Headings" in topic 3.2.5.
Instructions	Tells users what to do with the data in the panel. See "Instructions" in topic 3.2.4.
Message area	Reserved area of panel used to display messages to users. See Chapter 16, "Messages" in topic 3.11.
Panel area separator	Provides a visual boundary between adjoining panel areas. See "Panel Area Separators" in topic 3.2.3.
Panel ID	Identifies a specific panel. It may be used to identify the panel for maintenance or other purposes. See "Panel ID" in topic 3.2.1.
Panel title	Identifies either the purpose of the panel or the data in the panel. See "Panel Title" in topic 3.2.2.
Pop-up	A bordered area of the screen that extends the users' dialog with the underlying panel. See Chapter 14, "Pop-Ups" in topic 3.9.
Protected text	Protected information that can be viewed but not modified by users. See "Protected Text" in topic 3.2.8.
Scrolling information	Tells users that more information exists outside the visible panel area. There are three types: scrolling arrows, textual scrolling information, and textual scrolling location information. See "Scrolling Information" in topic 3.8.3.
Selection field	Presents a fixed, non-scrollable set of related choices from which users make selections. There are two basic types: Single-choice selection field: Users can select one choice or not make a selection. Multiple-choice selection field: Users can select any number of choices or not make any selection. See "Selection Elements" in topic 3.3.2.
Selection indicator	Reminds users which choices are selected. It may be a number, the slash symbol (/), or a country-designated character displayed in the choice entry field.
Selection list	Presents a potentially scrollable list of related choices from which users make selections. The list is typically variable in content or number but it also may be fixed. There are two basic types: Single-choice selection list: Users can select one choice or not make a selection. Multiple-choice selection list: Users can select any number of choices or not make any selection. See "Selection Elements" in topic 3.3.2.
Work area	The space between the panel title separator and the message area. It is the users' workspace and the focus of their attention.

3.1.3 Component Requirements by Model

Table 2 summarizes the Common User Access requirements, recommendations, and allowable options for user interface components in a nonprogrammable terminal (*NPT*) environment using the *entry* model, in a programmable workstation (*PWS*) environment using the *entry* model, and in an *NPT* environment using the *text subset* of the graphical model. This table must be used in conjunction with the rest of this book to design a CUA interface for your application. The abbreviations in the table are:

Fnd Fundamental to CUA -- required unconditionally

Fnd-C Fundamental to CUA under the conditions stated in the accompanying note--required conditionally

Rec Recommended by CUA

Rec-C Recommended by CUA under the conditions stated in the accompanying note

Opt Optional; if used, must follow any rules specified by CUA

Auto Automatically provided by the OS/2* software or hardware

HW Automatically provided by terminal hardware

-- Not applicable.

Note: Implementation of all the fundamental components of CUA will achieve a base level of consistency across applications. However, to achieve a significant level of consistency, applications should also implement the recommended components.

For recommended components, you should consider both the value of added consistency and the difficulty of implementation in a specific environment. For example, while BMS for the CICS/VS environment supports most of the recommended components, it is difficult for the application programmer to support user options, such as turning the function key area on and off and varying colors. In this example, the cost of implementing the recommended components may exceed the value gained.

Table 2. Common User Access Style Matrix

Interface Component	**NPT Entry**	**PWS Entry**	**NPT Text**	**Notes**
Presentation Style				
Single action/transaction style	Rec	Rec	--	
Object/action style	Opt	Opt	Fnd	
Action bar	--	--	Fnd	
Pull-downs	--	--	Fnd	
Show accelerator keys in pull-downs	--	--	Rec	
File (standard action bar choice)	--	--	Fnd-C	1
New, Open, Save, Save as, Print, and Exit (standard File pull-down choices)	--	--	Fnd-C	2
Edit (standard action bar choice)	--	--	Fnd-C	3
Undo, Mark, Cut, Copy, Paste, Unmark	--	--	Fnd-C	4

(standard Edit pull-down choices)				
Clear, Delete	--	--	Opt	
View (standard action bar choice)	--	--	Fnd-C	5
Options (standard action bar choice)	--	--	Fnd-C	6
Help (standard action bar choice)	--	--	Fnd-C	7
Windowing				
Full-screen primary window	Fnd	--	Rec	
Multiple movable windows, such as application, dialog, message	--	Auto	Opt	
Pop-ups for dialogs	Opt	Rec	Fnd	
Stationary	Rec-C	--	Rec	8
Movable	--	Auto	Opt	

Notes:

1. If the application works with stored objects.
2. If the File pull-down is supported.
3. If the application is a text editor or supports cut, copy, and paste.
4. If the Edit pull-down is supported.
5. If the application supports the display of objects in different formats.
6. If the application supports customization of objects.
7. If help is supported.
8. If pop-ups are supported.

Interaction Mechanisms

Keyboard	Fnd	Fnd	Fnd	
Replace mode is default	HW	--	HW	
Insert mode is default	--	Auto	--	
Mouse	--	Auto	--	

Panel Elements

Single-choice selection field	Fnd-C	Fnd-C	Fnd-C	9
Single-choice selection list	Fnd-C	Fnd-C	Fnd-C	10
Multiple-choice selection field	Fnd-C	Fnd-C	Fnd-C	11
Multiple-choice selection list	Fnd-C	Fnd-C	Fnd-C	12
Action list	Opt	Opt	Opt	
Action codes	Rec-C	Rec-C	Rec-C	13
Action entry fields (non-extendable)	Rec-C	Rec-C	Rec-C	14
Commands	Opt	Opt	Opt	

Extendable action entry fields	Rec-C	Rec-C	Rec-C	15
Type over column fields	Opt	Opt	Opt	
Panel ID	Opt	Opt	Opt	
Panel area separators	Fnd	Fnd	Fnd	
Panel title	Fnd-C	--	Fnd-C	16
Window title	--	Auto	--	
Instructions	Opt	Opt	Opt	
Field prompts	Rec	Rec	Rec	
Dot leaders . . .	Rec	--	Rec	
Column headings	Rec	Rec	Rec	
Entry fields	Fnd-C	Fnd-C	Fnd-C	17
Group headings	Rec	Rec	Rec	
Horizontally scrollable entry field	--	Auto	--	
Protected text	Rec	Rec	Rec	
Copyright statement	Fnd-C	Fnd-C	Fnd-C	18
About... pull-down choice	--	--	Rec	
Function key area	Fnd	Fnd	Fnd	

Notes:

9. If fixed, non-scrollable set of single-choice selections.
10. If variable or fixed, potentially scrollable set of single-choice selections.
11. If fixed, non-scrollable set of multiple-choice selections.
12. If variable or fixed, potentially scrollable set of multiple-choice selections.
13. If action lists are supported.
14. If action codes are supported but action list commands are not.
15. If action list commands are supported, or if action list commands and action codes are both supported in the same list.
16. Panel titles are optional in message pop-ups.
17. If user input is required by the application.
18. If required by corporate standards.

Selection Elements

Single-choice selection field				
Selection technique				
Number in choice entry field	Fnd	--	Fnd	
Point-and-select with cursor and pointer	--	Auto	--	
Mnemonics	--	Fnd	--	
Selection indicator				
Number in choice entry field	Fnd	--	Fnd	
Filled-in radio button	--	Auto	--	
Check mark (&check.)	--	Auto	--	

(pull-down only)				
Single-choice selection list				
Selection technique				
Slash character (/) and country-designated character in choice entry field	Fnd	--	Fnd	
Point-and-select with cursor and pointer	--	Auto	--	
Selection indicator				
Slash character (/) and country-designated character in choice entry field	Fnd	--	Fnd	
Inverse color in list box	--	Auto	--	
Multiple-choice selection field				
Selection technique				
Slash character (/) and country-designed character in choice entry field	Fnd	--	Fnd	
Point-and-select with pointer and cursor-and-spacebar	--	Auto	--	
Selection indicator				
Slash character (/) and country-designated character in choice entry field	Fnd	--	Fnd	
X in check box	--	Auto	--	
Check mark (&check.) (pull-down only)	--	Auto	--	
Multiple-choice selection list				
Selection technique				
Slash character (/) and country-designated character in choice entry field	Fnd	--	Fnd	
Point-and-select with pointer and cursor-and-spacebar	--	Auto	--	
Selection indicator				
Slash character (/) and				

country-designated character in choice entry field	Fnd	--	Fnd	
Inverse color in list box	--	Auto	--	

Messages

Message area	Fnd	--	Fnd	
Message pop-up	Fnd-C	Auto	Fnd-C	19
Message types				
Information	Fnd	Fnd	Fnd	
Warning	Fnd	Fnd	Fnd	
Action	Fnd	Fnd	Fnd	

Note:

19. If pop-ups are supported and user response to message is required.

Audible Feedback

Beep on warning and action messages	Rec	Rec	Rec	
Beep on invalid input	--	Auto	--	
Beep on invalid input on next host interrupt	Rec	--	Rec	

Commands

Command support	Rec	Rec	Rec	
Command area in primary window	Rec-C	Rec-C	Rec-C	20
Command area in pop-up	Opt	Opt	Opt	
Prompt support	Opt	Opt	Opt	

Note:

20. If significant use of system commands is expected.

Function Key Area

Function key area				
Invoke action by pressing key	Fnd	Fnd	Fnd	
Invoke action by cursor select	--	Auto	--	

Scrolling

Cursor-dependent	Fnd-C	--	Fnd-C	21
Cursor-independent	Fnd-C	Auto	Fnd-C	21
Cursor-driven (by arrow	--	Auto	--	

keys)					
Scrolling information	Fnd-C	Fnd-C	Fnd-C	22	
Scrolling arrows					
More:	Rec-C	--	Rec-C	23	
More: **< - + >**	Opt	--	Opt		
Textual scrolling information					
More..., Bottom	Opt	--	Opt		
Scroll bars	--	Auto	--		
Textual scrolling location information	Opt	Opt	Opt		

Notes:

21. Application must support either cursor-dependent or cursor-independent scrolling.
22. If panel area scrolls.
23. If it can be displayed on device.

Common Actions

Enter to process input	Fnd	Fnd	Fnd	
Help	Rec	Rec	Rec	
Refresh	Rec	Rec	Rec	
Prompt	Rec	Rec	Rec	
Command	Rec-C	Rec-C	Rec-C	24
Retrieve	Rec-C	Rec-C	Rec-C	24
Switch to action bar (Actions)	--	--	Fnd	
Display panel IDs	Fnd-C	Fnd-C	Fnd-C	25
Display keys	Fnd-C	Fnd-C	Fnd-C	26
Exit	Fnd	Fnd	Fnd	
Cancel	Fnd	Fnd	Fnd	
Switch window	--	Auto	--	
Backward	Fnd-C	Fnd-C	Fnd-C	27
Forward	Fnd-C	Fnd-C	Fnd-C	27
Left	Fnd-C	Fnd-C	Fnd-C	28
Right	Fnd-C	Fnd-C	Fnd-C	28

Notes:

24. If commands are supported.
25. If panel IDs are supported.
26. Do not provide if your application supports multiple sets of function keys.
27. If vertical scrolling is required.
28. If horizontal scrolling is required.

Help

Contextual help	Fnd-C	Fnd-C	Fnd-C	29

Extended help	Fnd-C	Fnd-C	Fnd-C	29
Help index	Rec-C	Rec-C	Rec-C	29
Search support	Opt	Opt	Opt	
Keys help	Rec-C	Rec-C	Rec-C	29
Reference phrase	Opt	Opt	Opt	
Help for help	Rec-C	Rec-C	Rec-C	29
Tutorial	Opt	Opt	Opt	
About...	--	--	Rec-C	29
Display in full panel	Opt	--	--	29
Display in pop-up	Rec-C	--	Fnd-C	29,30
Display in secondary window	--	Auto	--	

Notes:

29. If Help is supported.
30. Modeless pop-up is recommended.

User Profile Options

Color for each panel element type	Rec	Rec	Rec	
Turn panel IDs display on/off Default is *off*	Rec-C Rec	Rec-C Rec	Rec-C Rec	31
Turn function key area display on/off Default is *on* Default is *off*	Rec-C Rec --	Rec -- Auto	Rec-C Rec --	32
Set beep on/off Default is *on* Default is *off*	Rec Rec Opt	Rec Auto --	Rec Rec Opt	

Notes:

31. If panel IDs are supported.
32. Do not provide if your application supports multiple sets of function keys.

3.2 Chapter 7. Panel Elements

This chapter describes common panel elements and the general layout and location of each. These elements are:

- Panel ID
- Panel title
- Panel area separators
- Instructions
- Column headings and group headings
- Field prompts
- Descriptive text
- Protected text.

Other panel elements are used only with specific components or in specific conditions. They are:

- Action codes: see "Action Lists" in topic 3.3.3.
- Scrolling information: see "Scrolling Information" in topic 3.8.3.

Subtopics

3.2.1 Panel ID

The *panel ID* is an alphanumeric, character-string identifier. It is an application option to provide panel IDs. If they are provided, include them for all panels in the application. It is a user option to display the panel IDs.

Users can display all panel IDs in an application by requesting the Display panel IDs common action from an action bar pull-down, or by typing a command in a command area. You may assign this action to a function key that, alternately, turns the panel IDs on and off. The Common User Access recommends a default of off.

Layout: Place the panel ID left-justified on the same line as the panel title. Keep an obvious, visible separation between the panel ID and the panel title.

Content: The panel ID is an alphanumeric, character-string identifier that is unique within the application. It does not contain any blank characters. The panel ID may be either uppercase or mixed-case. Use whichever case you choose consistently throughout the application.

PICTURE 52

Figure 52. Panel ID. The panel ID ***Stat1*** in this entry model panel is left-justified on the same line as the panel title, ***Current Status***.

3.2.2 Panel Title

A *panel title* describes the information in the work area for users. All panels must have a panel title, including panels presented in pop-ups, except messages.

Layout: If the panel does not have an action bar, the title appears on the top line of the panel. If the panel has an action bar, the panel title appears on the next line below the action bar separator line.

Center the panel title.

Content: The panel title is one line of protected text. (For information about protected text, see "Protected Text" in topic 3.2.8.) The title may contain variable information, such as an object name, but it may not contain entry fields or selection fields. Use headline-style capitalization for panel titles. For applications created in English, capitalize the first and last words and all other words except articles, prepositions, coordinating conjunctions, and the word *to* in infinitives.

In primary windows, the panel title contains the application name and, if applicable, the object name. For example,

    Editor - My Document

In pop-ups, the panel title identifies the action of the pop-up. For example, if a pop-up results from selecting the action bar pull-down choice Open, the panel title for the pop-up would be *Open*.

PICTURE 53

Figure 53. Panel Title in Panel with Action Bar. For this text subset panel, the panel title ***Patient Services Update*** is centered on the line below the action bar separator. The blank separator line below the panel title distinguishes it from the work area.

3.2.3 Panel Area Separators

Panel area separators, as their name implies, distinguish adjoining areas in a panel. Use separators to define areas of a panel with different information.

Layout: Put a panel area separator:

- Between the action bar and the panel title
- Between the panel title and the work area
- Between different panel areas that scroll.

Content: Use a solid or dashed line as the separator between the action bar and the panel title.

You may use a blank line or a solid or dashed line as a separator between other panel areas.

Because the message area is usually blank, it may be used as the separator between the work area and the command area, or between the work area and the function key area if there is no command area.

The separator between the panel title and the work area may be a blank line or a line that contains scrolling information. Because scrolling information occupies only the right-hand portion of the line, the remaining blank portion provides visual separation between the panel title and the work area.

As an application option, you also may use a separator between other panel areas and elements, as needed, to distinguish clearly the areas or elements.

3.2.4 Instructions

Instructions tell users how to interact with a panel and how to continue with the application. The top of the work area should be used most often for instructions. Protected text may be used elsewhere in the panel to give users additional instructional information.

Usability testing of your panels will help you determine if you need to include instructions.

Layout: Place top instructions at the top of the work area below the panel title. Bottom instructions should be located near the bottom of the work area, that is, above the message area, command area, and function key area if any of these areas is being displayed.

Instructions are left-justified in the work area.

Content: Instructions are one or more lines of text in mixed case, following the rules for sentence-style capitalization.

Following are a few tips to help you write clear instructions:

Use the verb *type,* instead of *key, key in,* or *enter*.

> Type information.

Distinguish the action *select* from the action *Enter.* Select means to mark a choice; Enter means to send all selected choices to the computer for processing.

Use the verb *press* to tell users to initiate an action through the keyboard, such as initiating an action with a function key. For example,

> To display a list of valid choices, press the Prompt key.

Use the names of the actions in addition to or instead of the keys to which the actions are assigned.

> To display a list of valid choices, press the Prompt key (F4).

PICTURE 54

Figure 54. Panel with Instructions. The instructions in this entry model menu tell users that they can select only one of the values listed and remind them to press the Enter key to process their selection.

3.2.5 Headings

Headings identify columns and groups of related items. CUA defines two types of headings, column headings and group headings.

Both types of headings consist of text in mixed case. CUA recommends that you use headline-style capitalization for both types of headings. In applications created in English, capitalize the first and last words and all other words except articles, prepositions, coordinating conjunctions, and the *to* in infinitives.

Subtopics
3.2.5.1 Column Headings
3.2.5.2 Group Headings

3.2.5.1 Column Headings

Column headings identify columns of entry fields, selection fields, selection lists, action lists, and protected text when all the items in the column are the same type.

You should include some sort of identifier for every field or list unless it is the only one in the panel and the panel title identifies it.

General Layout: Column headings are located above the field or columns of fields they identify. If the information can be scrolled vertically, place column headings in a separate area that does not scroll, so that the headings do not scroll out of view. If the information scrolls horizontally, the headings scroll with it.

Layout Options for Numeric and Character Data: You may determine the layout of the column heading according to the type of data and whether the data is longer or shorter than the heading. Following are the layout options.

Fields containing numeric data -- The same options apply for both fixed-length and variable-length numeric data.

- If the field is longer than the heading, place the data right-aligned with the heading. For example,

```
  $Cost
1254.45
 875.99
2550.00
```

- If the field is shorter than the heading, center the data under the heading. For example,

```
Quantity
   21
   15
   36
```

Fields containing character (alphanumeric) data -- The options for character data depend on whether the length of the data is fixed or variable.

- If the field is fixed length, CUA recommends that you center the heading regardless of whether the data is longer or shorter than the heading. For example,

```
Rating          Code
  A1          AA2500A1
  A1          AA2600A1
  A2          AB3500A2
```

  However, you also may place the heading left-aligned over data that is longer than the heading.

- If the field is variable in length and longer than the heading, place the heading left-aligned. For example,

```
Description
Memo to J.P. Jones
Monthly accounting summary
Year end report
```

- If the field is variable in length and shorter than the heading, center the data under the heading. For example,

```
Item Color
  Green
  Yellow
  Blue
```

Column Headings for Groups of Columns: A single column heading may be used over multiple columns. If the heading is shorter than the width of the columns it applies to, center the heading over the columns and use a solid line on either side of the heading to indicate the span of the heading. For example,

```
  ----Authority----
Read  Add  Update
```

```
   X     X     X
   X     X     X
```

Hyphens may be used if a solid line is not available. For example,

```
    ----Authority----
    Read  Add  Update
     X     X     X
     X     X     X
```

Figure 55 shows an example of column headings in a scrollable panel.

PICTURE 55

Figure 55. Column headings in a Scrollable Panel. For this text subset panel, the column headings are ***Document, Subject,*** and ***Revised***. The panel contains more data than is visible so the data is presented in a panel area that users can scroll. The column headings are in a separate area from the data so they will remain visible as the data is scrolled. The message area, command area, and function key area are also separate from the scrollable area.

3.2.5.2 Group Headings

Group headings identify related groups of entry fields, selection fields, and protected text. You may use them concurrently with field prompts (see "Field Prompts" in topic 3.2.6) when group headings identify a group of fields and field prompts identify the individual fields.

Place a group heading above each group of fields. Place all group headings in a particular panel area left-aligned. Group headings should be indented when they follow instructions. Also, field prompts should be indented under the group headings.

If group headings are located in panel areas that scroll, the headings scroll with the items they identify.

PICTURE 56

Figure 56. Group Headings and Entry Field Prompts. The group headings ***Security Control, Printers,*** and ***Communication*** identify groups of fields. Notice that the group headings are indented to set them off from the instruction above them. Field prompts (for example, ***Bits*** and ***Parity*** under ***Communication***) identify the entry fields. The field prompts are indented under their respective group headings to make it clear that they are subordinate to the group headings.

3.2.6 Field Prompts

Field prompts identify selection fields, entry fields, and variable output information.

You may use field prompts concurrently with group headings when the group headings identify groups of fields and the field prompts identify individual fields within a group.

You should use a field prompt or other identifier for each field unless it is the only field in the panel and the panel title identifies it.

Layout: Field prompts are located to the left of the fields they identify. They are left-aligned.

If the field prompts follow group headings, the field prompts should be indented under the group headings. When group headings are not used, you should indent field prompts under instructions.

Use leader dots (. . .) to connect field prompts and fields, so users can easily move their eyes from one side of the screen to the other. When you use leader dots, use a minimum of two dots on a line and align the dots. You do not need leader dots for the longest field prompt in a group of aligned fields or for a panel with only one field.

Field prompts for variable output information also should be followed by a string of leader dots. Replace the last leader dot with a colon (:) to indicate that what follows is protected text. You do not need leader dots for the longest field prompt, but you must have the colon. The colons are lined up vertically, just as leader dots are.

Content: Field prompts use normal sentence-capitalization.

PICTURE 57

Figure 57. Field Prompts in a Scrollable Panel. The field prompts are ***Document name, Type style, Number of copies, Left margin, Start page number, One or both sides,*** and ***Keep print file***. The field prompts, their associated strings of leader dots, and the entry fields and selection fields they identify are all part of the scrollable area. When the panel is scrolled, they all scroll together.

PICTURE 58

Figure 58. Field Prompts for Protected Text. The field prompts ***Document name*** and ***Type style*** identify protected text that users cannot select or modify. The colons at the end of the strings of leader dots and the absence of entry field underscore delimiters indicate to users that this information is protected. The field prompts ***Number of copies, Left margin,*** and ***Start page number*** identify entry fields in which users can type values. The field prompts ***One or both sides, Keep print file,*** and ***Classification*** identify selection fields from which users can make a choice.

3.2.7 Descriptive Text

Where it would be useful to users, you may include additional descriptive information about the entry field, in addition to the field prompt that identifies the field. Such information is *descriptive text*.

Layout: Whereas the field prompt precedes the entry field, descriptive text is placed after the end of the field. Allow sufficient space between the end of the field and the beginning of the descriptive text to provide visual separation. Place the descriptive text for all entry fields in the same panel area left-aligned. If the descriptive text does not fit on the same line as the entry field, it should be continued on the next line under the descriptive text on the previous line.

Content: The descriptive text should be brief, protected text. It should provide information about which values can be typed into the entry field. If the list is small, the actual values may be displayed as descriptive text, separated by commas and blanks, as follows:

```
Graph type . . .  ___________   Scatter plot, Bar, Line, Pie
```

A range of values also may be indicated as descriptive text. For example:

```
Copies . . . . .  __   1-99
```

Use capitalization rules that are consistent with the type of information presented in the descriptive text. For example, if you are listing a set of separate values, capitalize the first letter of each value in the list. You also may use all uppercase for keywords or special values that must be typed exactly as shown. If the descriptive text is in the form of a phrase, use sentence-style capitalization.

PICTURE 59

Figure 59. Entry Field with Descriptive Text. The descriptive text, ***Number of bushels ordered***, tells users the kind of information that can be typed in the the entry field that is identified by the field prompt ***Bushels***. The field prompt ***Grain elevator site*** identifies another entry field, and the field prompt ***Grain*** identifies a selection field.

3.2.8 Protected Text

Panels may have areas that contain text that only can be viewed, not selected or modified. Such information is *protected text*. Protected text may be text in a file, numeric data, help information, a message, or variable output information. An example of variable output information is shown in Figure 60. When protected text is used for variable output information, follow the color and emphasis technique that is assigned for variable output information. When protected text is used for other information, such as for help panels or in messages, follow the technique for normal text. Figure 61 shows an example of a panel with normal text. You can find the color and emphasis techniques for protected text in Table 8 in topic APPENDIX1.2.1.

Layout and Content: Protected text may be located in a part of the work area, or it may occupy the entire work area.

Format protected text in a way that is most appropriate for the type of information. For example, numeric data and groups of data that are the same type might be presented best in a tabular format.

If a tabular format is used, provide column headings and align the data as specified in "Headings" in topic 3.2.5. Also, follow all the relevant formatting rules that are specified in "Selection Elements" in topic 3.3.2 and "Action Lists" in topic 3.3.3.

If the information is textual, you might make it easier to read by not aligning the right margin; leave it ragged-right. To make the information easier to scan, use short paragraphs and lists, rather than long, continuous strings of text.

CUA defines additional requirements and recommendations for help information and messages. For more details about help information, see Chapter 15, "Help" in topic 3.10. For more details about messages, see Chapter 16, "Messages" in topic 3.11.

If all the information will not fit within the area, make the area scrollable. As with any other scrollable area, you must provide the scrolling actions and scrolling information defined in Chapter 13, "Scrolling Panel Areas" in topic 3.8.

How Users Interact with Protected Text: Because the information is protected, users cannot type over it or change it. If the information is too large to fit within the panel, users can scroll the panel.

If the information is presented in a pop-up or a full panel, users can remove the information by removing the pop-up or panel, using common actions, such as Enter, Exit, or Cancel.

PICTURE 60

Figure 60. Panel with Protected Text. The protected text is presented using field prompts in a format similar to the field prompts used with entry fields. However, the information is protected, as indicated by the colons at the end of the leader dots and by the absence of the underscore delimiters that indicate entry fields.

PICTURE 61

Figure 61. Another Panel with Protected Text. Protected text occupies the entire work area of this help panel. Notice the way the information is split into small units to make it easier to scan.

3.3 Chapter 8. Entry and Selection

Entry and selection are the two basic techniques by which users communicate with any computer application.

By *entry*, the Common User Access means that users type character or numeric information, using the keyboard. The areas of a panel into which users type this information are known as *entry fields*.

By *selection*, CUA means that users choose from a group of related *choices* that are presented in the panel. Choices are usually words in nonprogrammable terminal applications. The area of the panel where the group of choices is presented and where users indicate their selection from among these choices is known as a *selection field, selection list*, or *action list*.

Note: The word *select* means to pick a choice. It is not synonymous with *Enter*, which is a CUA common action that means that the panel containing the selected choice is submitted to the computer for processing. This is an important distinction to remember as you read this book.

If a panel contains several entry fields, selection fields, selection lists, action lists, or a combination of these, users move to the field they want to type into or to the choice they want to select by moving the cursor to the desired field or choice.

Subtopics

3.3.1 Entry Fields

An entry field is an area of a panel into which users type information. The location and layout of entry fields are determined by application designers, based on guidelines in this chapter. These guidelines apply only to entry fields in which information is typed from left-to-right and is limited to one line.

Subtopics
3.3.1.1 Entry Field Layout
3.3.1.2 What Happens When an Invalid Value Is Entered

3.3.1.1 Entry Field Layout

When a panel first appears, an entry field may be displayed either with or without an initial value.

Delimiters: You should provide entry field delimiters to visually indicate the length of the field. For display devices that support underscoring, use the underscore attribute as the delimiter to show all available entry positions. For example,

```
Name . . .   _________________________
```

shows an entry field in which users type a person's name.

For displays that do not support the underscore attribute, it is an application option to use the underscore character. When an application uses the underscore character, the characters typed into the entry field replace the underscores.

Identifying Entry Fields: Application designers typically identify entry fields on the panel with descriptive panel elements, such as *field prompts*, *column headings*, *or descriptive text* to help users know what kind of information entry fields can contain. For details about the use of descriptive panel elements, see Chapter 7, "Panel Elements" in topic 3.2.

When optional entry fields are displayed with entry fields that must have a value before the panel can be processed successfully, distinguish the required entry fields from the optional entry fields, whenever possible. This may be done by using descriptive text or emphasized input for the required entry fields.

PICTURE 62

Figure 62. Panel with Entry Fields. Underscore delimiters indicate the length of the entry fields. Each entry field is only long enough to allow entry of the largest value accepted in the field.

3.3.1.2 What Happens When an Invalid Value Is Entered

If users type an invalid value into an entry field and press the Enter key, the value should be indicated by the cursor position or with error emphasis, and a message should appear. The message indicates that the value is invalid. As an application option, you may provide help for the message. The help panel may describe the valid values for the entry field if they were not listed in the descriptive text for the entry field or in the message text. Depending on the characteristics of the terminal, a beep may or may not sound.

3.3.2 Selection Elements

A selection element is a group of related choices from which users can make one or more selections. CUA defines four types of selection elements:

Single-choice selection fields

Single-choice selection lists

Multiple-choice selection fields

Multiple-choice selection lists.

Selection *fields* are fixed in content and number of choices and are not scrollable. *Selection lists* typically vary in content or number of choices and are potentially scrollable.

Selection fields and selection lists allow users to select one or more choices or not make any selection, as follows:

Single-choice selection: Users can select only one choice or not make a selection.

Multiple-choice selection: Users can select any number of choices or not make any selection.

Users select one or more choices by typing a number or character into the choice entry field associated with the desired choice or choices.

After users make their selection, users submit the panel for processing, usually by pressing the Enter key.

Selection Acknowledgement: When users select a choice, the application should give some visual acknowledgement that the choice is selected. This feedback is provided by characters or symbols placed in choice entry fields, referred to as *selection indicators*. Visual acknowledgement may be different, depending on the type of selection element. To determine the correct visual acknowledgment, refer to the information about each type of selection element.

Selection Element Content: Selection elements may contain different types of choices, depending on where they are used. Selection elements located in the work area typically contain objects, such as people's names, or options, such as marital status (married, single, divorced), from which users select. Selection fields that appear in pull-downs typically contain actions, such as Open and Save. For entry model applications, however, selection elements in the work area also may contain actions because these applications do not support an action bar. When actions and objects both appear in selection elements in the work area, do not mix actions and objects in the same selection element.

Choice Capitalization: Choices in selection elements should be a single word or a phrase. Capitalize the first letter of the choice. The exceptions to this rule are:

When the choice contains an acronym or abbreviation that is usually capitalized, use accepted capitalization.
When the choice contains a proper noun that contains a capital letter other than the first letter, use the accepted capitalization; for example, key engravings, such as *Page Up*, and some proper nouns, such as MacMurray.

These rules for capitalization apply only to English language text.

Subtopics

3.3.2.1 Single-Choice Selection Fields

A *single-choice selection field* contains a fixed number of choices from which users can select only one choice or not make a selection.

The choices in single-choice selection fields may be formatted in two ways: vertically in a left-aligned column, or horizontally on the same line.

The space between choices may be increased to group related choices.

Selection fields are also used in action bar pull-downs. For information on layout and content of selection fields in pull-downs, see "Action Bar Pull-Down Content" in topic 3.5.5.

A single-choice selection field has an associated entry field into which users type a number to select a choice. This entry field is known as a *choice entry field*. When the single-choice selection field is first displayed, the choice entry field may contain a default value.

The choice entry field is located to the left of the top, left choice and on the same line. It has as many character positions as needed to accommodate the largest number assigned to a choice in the selection field. For example, if the selection field contains nine or fewer choices, the entry field is one character long. If the selection field contains 10 to 99 choices, the entry field is two characters long.

Provide delimiters for the choice entry field, using the underscore attribute. When the underscore attribute is not available, use the underscore character.

Users select a choice in a single-choice selection field by typing the number of the desired choice into the choice entry field. Users de-select a choice by typing a number over, or deleting, the number in the choice entry field.

Users can position the cursor on a choice using the arrow keys, so they can request Help.

A single-choice selection field may be identified with:

- A field prompt
- A panel title, when the panel does not contain any other selection elements or any entry fields.

Subtopics
3.3.2.1.1 Numbering Choices

3.3.2.1.1 Numbering Choices

Choices in single-choice selection fields are always numbered. You should number the choices sequentially and consecutively in each field, beginning with the number one. Use the format: digit, period, space, choice text. When it is important to keep the same number assigned to the same choice in different selection fields and panels, you may have non-consecutive numbering. For example, a *9* could be used for *logoff* in every panel where the logoff choice is available. When numbers are not assigned consecutively, CUA recommends that you leave one blank line between the non-consecutive numbers. For example:

```
    _ 1. Edit
      2. Print

      9. Logoff
```

If the choices are numbers, CUA recommends that you place text between the numbers and choices. For example:

```
    _ 1. Room 13
      2. Room 18
      3. Room 30
```

If you cannot use text in this way, CUA recommends that you use words to express the number choices. For example:

```
    _ 1. Thirteen
      2. Eighteen
      3. Thirty
```

PICTURE 63

Figure 63. Single-Choice Selection Field. Users can select only one choice. Users make a selection by typing the number of the desired choice in the choice entry field to the left of the first choice.

3.3.2.2 Single-Choice Selection Lists

A *single-choice selection list* contains a potentially scrollable group of choices, from which users can select only one choice or not make a selection. Selection lists are typically variable in content or number, but they also may be fixed in both content and number.

Single-choice selection lists do not support numbers. The choices in the list are formatted vertically in a left-aligned column. A one-character entry field precedes each choice; the entry field contains the period character (.). Scrolling information appears on the line above the heading. If the list is not currently scrollable, scrolling information is not displayed. If the panel contains elements other than the selection list, you may use horizontal and vertical separators to provide visual separation between the list and other selection elements or entry fields in the panel.

Users select a choice in a single-choice selection list by moving the cursor to the desired choice and typing a slash character (/) or a country-designated character over the period character (.). Users de-select a choice by typing a blank over, or deleting, the slash character or a country-designated character. The period character preceding the de-selected choice is re-displayed after the next host interrupt, for example, after a scrolling action.

Users move from choice to choice using the arrow or Tab keys. Users scroll the list using the Backward and Forward actions (F7 and F8) while the cursor is in the list.

A single-choice selection list may be identified with:

- A column heading
- A panel title, when the panel does not contain any other selection elements or any entry fields.

PICTURE 64

Figure 64. Single-Choice Selection List. This pop-up, which contains a single-choice selection list, was displayed when users requested the ***Prompt*** action (see Chapter 9, "Prompt" in topic 3.4). Users select a choice in this list by moving the cursor to the name ***Carter, Judith*** and typing a slash character over the period. When users press the Enter key, the pop-up with the single-choice selection list is removed and the selected name is placed into the ***Patient name*** entry field in the underlying panel.

3.3.2.3 Multiple-Choice Selection Fields and Lists

A multiple-choice selection field or list contains a group of choices from which users can select any number of choices or not make any selection. *Multiple-choice selection fields* contain a fixed number of choices. *Multiple-choice selection lists* are potentially scrollable and typically vary in content or number of choices, but they also may be fixed in content and number.

The layout of a multiple-choice selection field and a multiple-choice selection list is similar. Choices in multiple-choice selection fields and lists are not numbered. Each choice in multiple-choice selection fields and lists is preceded by a one-character choice entry field. Each choice entry field has delimiters, consisting of underscore attributes. When the underscore attribute is not available, underscore characters are used.

Users select choices by typing a slash character (/) or country-designated character into the choice entry field that precedes each choice they want to select. Users de-select choices by typing a blank over, or deleting the slashes (or country-designated characters) in the choice entry fields.

When the field is initially displayed, default choices may be indicated by displaying slashes or country-designated characters in the choice entry fields.

A multiple-choice selection list differs from a multiple-choice selection field in that it has the ability to scroll. When a multiple-choice selection list is in its non-scrollable form, it looks identical to a multiple-choice selection field. However, when a multiple-choice selection list becomes scrollable, scrolling information is displayed on the line above and to the right of the column heading. Optional horizontal and vertical separators may be used to provide visual separation between the multiple-choice selection list and other selection elements and entry fields on the panel.

Because multiple-choice selection fields are fixed in content, they may be formatted vertically in a left-aligned column or horizontally on the same line. Multiple-choice selection lists are formatted vertically because they are potentially scrollable.

Users move from choice to choice in multiple-choice fields and lists using the arrow and Tab keys. Users scroll multiple-choice selection lists using the Backward and Forward actions (F7 and F8) while the cursor is in the list.

A multiple-choice selection field or list may be identified with one or more of the following:

- For a selection field, a field prompt placed to the left of the top, left choice entry field
- For a selection list, a column heading
- A panel title, when the panel does not contain any other selection elements or any entry fields.

PICTURE 65

Figure 65. Multiple-Choice Selection Field. This multiple-choice selection field allows users to select more than one choice, as stated in the instructions. A choice entry field appears in front of every choice. The slash characters (/) in the entry fields that precede ***Underline*** and ***Section numbers*** indicate that those choices have been selected.

PICTURE 66

Figure 66. Panel with a Multiple-Choice Selection List. This multiple-choice selection list looks similar to the multiple-choice selection field in Figure 65 because each choice is preceded by a choice entry field. However, the multiple-choice selection list contains scrolling information in the top, right corner, to indicate that it is scrollable.

3.3.2.4 Selection Acknowledgement

Selection acknowledgement varies, depending on the type of selection element:

In single-choice selection fields, the number that users type in the choice entry field is the selection indicator.

In single-choice selection lists and multiple-choice selection fields and lists, a slash or country-designated character in the choice entry field preceding the selected choice is the selection indicator.

3.3.2.5 Selection Element Initial Conditions

The *initial condition* of a selection field or selection list is its appearance when it is first displayed on the screen.

Initial Cursor Placement: The initial placement of the cursor is always to the left of the first displayed choice in the field.

- In single-choice selection fields, the cursor appears left-justified in the choice entry field.
- In single-choice selection lists and multiple-choice selection fields and lists, the cursor appears in the choice entry field of the first choice.

Default Choices: CUA allows the optional use of default choices for all selection fields and lists. When a panel is displayed initially, *default choices* are displayed as selected choices; the default choices are indicated by appropriate selection indicators.

When it is possible to predict in a certain circumstance which choice users often want to select, the presentation of default choices can be helpful. When the default choice is the users' choice, users can proceed immediately to the next step in the dialog.

The default choice of a selection field or list is one of the following:

- The choice that was selected and processed during a previous appearance of the field or list
- A choice determined by the application to be the most appropriate under the current circumstances.

Default choices are not recommended for selection lists if all the default choices cannot be displayed in the visible panel area.

3.3.2.6 Selection Element Unavailable Emphasis

A choice is unavailable if the application does not allow users to select it. For example, if a word processor requires that users paginate a document before selecting the *Best quality print* choice and if users do not paginate the document, the *Best quality print* choice is unavailable. However, other print choices might be available.

Do not display unavailable choices in selection lists. Do not maintain space in the list where an unavailable choice might appear when it is available.

Because selection fields are fixed in content, unavailable choices are always displayed. Therefore, you must let users know when choices are unavailable for selection, by using emphasis and highlighting. See Appendix B, "Color and Emphasis Table" in topic APPENDIX1.2 for the correct unavailable emphasis for your display device.

What Happens When an Unavailable Choice is Selected: When users select an unavailable choice and explicitly Enter the panel, a message appears telling users that that choice is unavailable. Also, the message reminds users that they can request help for the choice. The help panel for the choice should tell users the conditions that make the choice unavailable.

Unauthorized Choices: Do not show users choices they are not authorized to select. Do not confuse *unavailable* with *unauthorized.* Some applications provide several levels of functions for different categories of users. Users in one category may not be authorized to use functions at a certain level. Choices that are connected to this *off-limits function* are considered unauthorized by CUA, not unavailable. Such choices, therefore, are not displayed.

3.3.3 Action Lists

An *action list* displays a list of objects, each with an associated entry field, called an *action entry field*. Users can specify individually an action to be performed on each object in the list. Specifying an action automatically selects the object. The specified action can be different for each object. For example, in an application that keeps track of documents, users might use an action list to specify actions for three documents: an archive action for one, a print action for another, and a delete action for the third. All three actions will take place when the Enter key is pressed.

An action list is similar to a multiple-choice selection list. Both allow the selection of multiple objects; but a multiple-choice selection list only allows a single action to be performed on the selected objects.

Subtopics

3.3.3.1 Action List General Layout

The objects in an action list are arranged in columns.

Column headings are required. The heading above the action entry fields should be *Action* or *Option*. You may use the abbreviations *Act* and *Opt* for these headings. Provide enough space between the column heading of the action entry field and a preceding or following column heading to ensure visual separation. For example:

```
   Action  Document Name
     __    Letter1
     __    Memo1
     __    Note1
```

Action codes are recommended. *Action codes* are alphabetic or numeric representations of actions. If action codes are supported, an instruction appears above the action list. The instruction shows the action codes and the actions they represent. The action codes and actions should be displayed in one or more lines, from left-to-right and top-to-bottom. You may align the action codes vertically, as shown in the following example:

```
  Type an action code next to each desired choice; then press Enter.
      1=New     2=Open      3=Copy     4=List
      5=Print
```

Action list choices are not numbered.

The action entry field for each choice appears on the same line as the choice.

PICTURE 67

Figure 67. Action List with Action Codes. Users are instructed to type an action code into the action entry field that precedes the desired object. The action codes and actions (for example, ***3=Copy***) are part of the instructions.

3.3.3.2 Action List Interaction

CUA defines three techniques to specify actions to be performed on objects in an action list:

Action codes: Users type an action code into the action entry field associated with each object that is to be acted upon. CUA recommends that your application support the use of action codes.

In text subset applications, users also can select choices by typing a slash (/) or country-designated character in the action entry field. Users then switch to the action bar and request a pull-down containing the desired choice. The requested pull-down action is applied to all selected choices. This is the same interaction technique used in multiple-choice selection fields and lists.

Commands: Users type a command in the action entry field associated with each object that is to be acted upon. Support of this technique is optional. An action list may support commands alone or with either of the other interaction techniques.

Type over: Users type over the name or characteristics of the object displayed on the screen to change the name or characteristics. For example, to change the name of a document on a disk, users type the new document name over the existing name on the screen. When the application detects the change, it changes the document name on the disk. Support of type over is optional.

When users type an action code or command in an action entry field, the object associated with the action entry field is selected. To de-select an object, users type blanks over the characters in the action entry field, or delete the characters.

You should allow users to type an equal symbol (=) in an action entry field to indicate that the last action or command specified in a previous action entry field is to be applied also to each choice that is next to the equal symbol. In the following example, the Print action (**print**) would be applied to **Letter1, Note1**, and **Memo2**.

```
print      Letter1
           Memo 1
=          Note1
=          Memo2
           Letter 2
           Note 2
```

3.3.3.3 Action Codes

An action code should be a single number or letter. Make action codes in an action list either all alphabetic or all numeric. The format for presenting numeric and alphabetic action codes and their commands is shown in the following two examples:

```
Type one or more action codes; then press Enter.
  1=New     2=Open      3=Copy     4=List
  5=Print

Type one or more action codes; then press Enter.
  N=New     O=Open      C=Copy     L=List
  P=Print
```

When using numeric action codes, you may skip numbers to keep the same action code assigned to the same command throughout your application. In the following example, **Print** has an action code of **5** even though there is no command assigned to the number 4.

```
1=New  2=Open  3=Copy  5=Print
```

3.3.3.4 Action Entry Fields

CUA recommends that you place the action entry fields to the left of the object names, as illustrated in Figure 67 in topic 3.3.3.1, if your application supports

Only action codes or
Action codes and short command names without parameters (for example, Print and Discard).

Where users can type action codes or short commands, make the action entry fields long enough to contain the longest of these action codes or short commands.

If your application supports commands in an action list, CUA recommends that you make the action entry fields extendable. *Extendable action entry fields* allow users to type long command names or command names with parameters. Users must begin typing the command name in the visible portion of the action entry field. They can continue typing beyond the end of it, typing over other text on the line. This option is illustrated in Figure 68.

If your application supports extendable action entry fields, CUA recommends that you place the action entry fields to the right of the object names. This prevents users from obscuring the object names, which could cause confusion.

PICTURE 68

Figure 68. Action List with Extendable Action Entry Fields. Users type commands and parameters in the ***Action*** entry field and continue typing over the text that is to the right of the action entry field. Notice that the ***Action*** entry field is located to the right of the choice names so that the commands do not cover up the names.

If your application supports both type over and extendable action entry fields, the application must distinguish between the use of the type over technique and typing over displayed information during the use of extendable action entry fields.

Entry field underscores are optional for extendable action entry fields.

As an application option, you may include entry fields on the first line below the column headings, so users can type the name of an object that does not appear in the currently displayed list. The object may be a new object or an object in a part of the list that is not currently visible. You may include entry fields for several additional columns when such entry fields are necessary for users to adequately specify the object for the application. Provide an action entry field for these entry fields. This set of entry fields is in the non-scrollable panel area that contains the instructions and column headings. The action list in Figure 67 in topic 3.3.3.1 allows users to create a new document using the entry fields on the first line below the column headings **Document** and **Subject**. If an application supports entry fields on the first line below the column heading and the application is migrated to the programmable workstation, you must re-code the action list support.

In the following example of an action list, users can create a new file by typing a **1** in the **Action** entry field, the name of the file they are creating, **SALES87**, in the **File Name** entry field, and the subject of the file, **Net 1987 sales**, in the **Subject** entry field.

Type an action code next to each desired choice; then press Enter.
1=New 2=Open 3=Copy 5=Print

Action	**File Name**	**Subject**	**Last Changed**
1	SALES87	Net 1987 sales	
_	FINANCE1	Financial report	July 14
_	BUDGET13	Draft budget 13	May 4

3.3.3.5 Positioning the Cursor in Action Lists

Locate the cursor initially in the top, left entry field in an action list.

After the list has been processed and re-displayed, the cursor should appear next to the last choice that users selected in the panel, unless that choice is no longer visible. In that case, the cursor should be in the first entry field in the panel.

When the application detects errors, the cursor should be in the first position of the first entry field that is in error.

3.3.3.6 Action List Processing

The action codes and commands in action lists are processed only when users press the Enter key. No other action, such as scrolling, should cause the panel to be processed. Once entered, actions are performed in the order in which they appear in the list.

Some actions, such as Delete, may require user confirmation before processing. You may group these actions together on a separate panel to display them to users for confirmation. Users confirm the actions as a group, but each action is performed in the order in which it appears in the panel.

As an application option, users may interrupt action list processing. Your application determines how this is done.

If requested actions change the list of choices on the panel, the changes should appear the first time the panel is re-displayed by either the application or the users. For example, if users request an action to rename a file, the file will have its new name the first time the panel is re-displayed after the rename action has been processed.

If the application detects processing errors, processing stops at the point where the first error occurred. The cursor is positioned at the beginning of the action entry field that contains the error. An action message is displayed that tells users what to do.

When the application has processed all the actions, the action list is re-displayed with the same view users had when they requested the actions. All changes resulting from the actions are shown, to let users know that the actions were processed. Users can then request other actions.

When the application has processed the actions and the action list is re-displayed, the action entry fields of all processed choices contain an asterisk (*) to show users what objects were processed. The processed objects are no longer selected. The asterisk replaces the first character in the field.

The Refresh action updates the object list and erases the asterisks next to the previously processed choices. The list should be re-displayed with the same view users had when they requested the Refresh action.

3.3.4 Summary of Selection Element and Action List Characteristics

The following table uses a simple example to illustrate differences in the appearance of and interaction with single-choice selection fields and lists, multiple-choice selection fields and lists, and action lists.

Table 3. Examples of Selection Fields, Selection Lists, and Action Lists

Type	Examples and Comments
Single-Choice Selection Field	*3* 1. Chocolate 2. Strawberry 3. Vanilla 4. Butterscotch To select **Vanilla**, users type a **3** into the underscored choice entry field that precedes the first choice.
Single-Choice Selection List	. Chocolate . Strawberry / Vanilla . Butterscotch To select **Vanilla**, users move the cursor to **Vanilla** and type a slash or country-designated character over the period in the choice entry field.
Multiple-Choice Selection Field or Selection List	/ Chocolate _ Strawberry / Vanilla _ Butterscotch To select both **Chocolate** and **Vanilla**, users first move the cursor to **Chocolate** and type a slash character (/) or country-designated character into the choice entry field that precedes **Chocolate**. Users then select **Vanilla**, using the same technique.
Action List	A=Add D=Discard **Flavor** _ Chocolate _ Strawberry *a* Vanilla *d* Butterscotch This list may look a lot like the multiple-choice selection field in the previous example. But, instead of a /, users type an action code into the entry field. The action codes are displayed above the list. Users can type a different action code for each item. An alternate form of this type of selection list places the action entry fields *after* the items and uses no underscores for the entry fields, as shown below. **Flavor** **Action** **Calories** **Cost** **Rank** Chocolate 65 .15 1 Strawberry 33 .13 6 Vanilla order 16 gallons Butterscotch 57 .12 4 In this example, the application may allow users to continue typing the action and any parameters it may have over the data in the fields that follow the action entry field.

3.4 Chapter 9. Prompt

The *Prompt* action allows users to fill in entry fields by selecting from lists of values that are valid for those fields. It can save time for users and reduce the chance of typing errors.

When the list of valid values for an entry field is static *and* very short, it may appear as descriptive text next to the entry field. For information about descriptive text, see "Descriptive Text" in topic 3.2.7. However, if the list of valid values is too long for descriptive text or the list is determined dynamically when the application is run, Prompt should be provided. For example, you might use Prompt to list the printers currently available on the system when users must enter the name of a printer in an entry field.

Prompt is optional for the command area. For other entry fields, you should provide Prompt when the application *knows* the possible entries.

Prompt is for use with application-specific field types. It is not intended to be a substitute for the general-purpose directory and file list capabilities of the operating system.

To use Prompt, users place the cursor on the entry field for which they want a list of possible entries. When they request Prompt, by pressing the F4 key, a *prompt list pop-up* appears, containing a single-choice or multiple-choice selection field or list.

When users have selected what they want from the pop-up, the pop-up disappears. The choice text is placed into the entry field as though users had typed it there. Prompt gives users the opportunity of recognizing and selecting the choices they want, rather than having to remember all the choices and typing the desired ones.

The size of the pop-up is an application design decision. For *entry model* applications, a full-screen panel may be used if pop-ups are not available.

Users can cancel the prompt list pop-up without selecting a choice. Canceling the pop-up has no effect on the entry field.

If users request Prompt when the cursor is not on an entry field or is on an entry field that does not support the Prompt function, a message appears indicating that the Prompt function is not available. A *beep* may sound also. The message is removed on the next user action.

Allow users to request Prompt when they have entered an invalid value, so they can select a valid value.

As an application option, you may allow users to request Prompt for commands when the cursor is in the command area. For more information about Prompt for the command area, see Chapter 11, "Command Area" in topic 3.6.

Subtopics
3.4.1 Prompt Indicator
3.4.2 Tailoring the Prompt List

3.4.1 Prompt Indicator

For entry fields *other than the command area*, if Prompt is available, a prompt indicator, the plus sign (+), is shown at the end of the field. Instead of the prompt indicator, you may place descriptive text, such as **Press F4 for Prompt** at the right of the entry field.

No prompt indicator follows the command area.

PICTURE 69

Figure 69. Entry Field with Prompt Action. Users place the cursor on the entry field and press F4 to request the Prompt action.

PICTURE 70

Figure 70. Prompt List Pop-Up. This pop-up results when users request Prompt from the panel shown in Figure 69. The list allows users to find the name of a patient without having to remember it. Users select ***Carter, Judith*** and see the panel in Figure 71.

PICTURE 71

Figure 71. Users' Prompt Choice Appears in the Patient Name Entry Field

If a column of entry fields is provided to allow multiple entries, the prompt indicator may be placed at the end of the column heading, rather than after each field.

```
    Patient Name +
    _______________
    _______________
    _______________
    _______________
    _______________
```

3.4.2 Tailoring the Prompt List

As an application option, you may allow users to qualify, or *tailor*, the list of valid choices in the selection field or list by typing a search string in the entry field for which they are requesting Prompt. The string may include global search characters. The global search characters are *?* and ***.

? in a search string allows a single character to occupy that position in the string.

* in a search string allows any string of characters to occupy that position.

If users can type both double-byte and single-byte characters in an entry field, use the ? to represent any one single-byte or double-byte character occupying that position. Use the * to represent a string of any single-byte or double-byte characters. If users can type only double-byte characters, the double-byte ? represents any one double-byte character occupying that position. The double-byte * represents any string of double-byte characters.

The choices that appear in the selection list will be specific to the search string.

PICTURE 72

Figure 72. Qualifying a Search of Possible Valid Entry Field Values. Users type ***Martin**** in the entry field to limit the list of patients to those whose last names begin with ***Martin***, when Prompt is requested.

The following figure shows the various search strings and identifies what the results would be if Prompt were performed with those strings in the panel shown in Figure 70 in topic 3.4.1.

Patient name . . . +

The Prompt action lists all items, as in Figure 70 in topic 3.4.1.

Patient name . . . * +

The Prompt action lists all items, as in Figure 70 in topic 3.4.1.

Patient name . . . *s +

The Prompt action lists all items that end in **s**.
In Figure 70 in topic 3.4.1, only **Edwards** and **Mathews** would be listed.

Patient name . . . b* +

The Prompt action lists all items that begin with **B**.
In Figure 70 in topic 3.4.1, only **Baker** and **Basker** would be listed.

Patient name . . . ? +

The Prompt action lists all items that are single characters.
In Figure 70 in topic 3.4.1, no items would be listed.

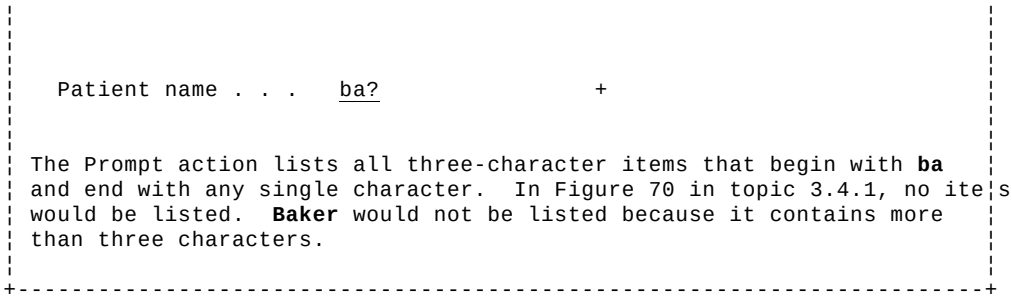


Figure 73. Strings Typed into Entry Fields before Prompt Is Requested

3.5 Chapter 10. Action Bar and Pull-Downs

The *action bar* is the panel element at the top of a panel that consists of a list of choices that represent groups of related actions that users can request. A group of actions appears in a *pull-down* when users request an action bar choice. Pull-downs are located immediately below the action bar.

The actions typically affect information displayed in the work area or in some way control the users' dialogs with the application.

Figure 74 illustrates an action bar and one of its pull-downs.

PICTURE 73

Figure 74. Action Bar and Pull-Down. The action bar contains the choices ***File, Edit, View, Options,*** and ***Help.*** The ***View*** choice was selected, causing its pull-down to appear immediately below. The choices in the pull-down are ***Since admission, Past 7 days, Past 72 hours,*** and ***Past 24 hours.***

When to Use: All applications based on the text subset must have an action bar on every panel in the primary window. Pop-ups do not have action bars.

Do not use an action bar in applications based on the entry model.

Subtopics

3.5.1 Action Bar Layout

The action bar is located at the top of a panel, as illustrated in Figure 75.

PICTURE 74

Figure 75. Action Bar Location. The action bar is located at the top of the panel and immediately above the panel title, ***Patient Services Update.***

The action bar stretches across the full width of a panel, regardless of the number of items in the action bar. Choices are listed horizontally on one or more lines with the same number of blanks between all choices. A blank precedes the text for each choice, so users can position the cursor where it is seen more easily.

Position the first choice far enough in from the left edge of the panel so you can properly position the pull-down below the choice, as defined in "Action Bar Pull-Down Layout" in topic 3.5.4.

Separate the action bar from the panel areas below it using a solid or dashed line, as shown in Figure 75.

3.5.2 Action Bar Content

The action bar contains choices in the form of single or multiple words. Use single-word choices when you can. Use multiple-word choices only when one word cannot describe an action category.

Capitalize the first letter of the choice. The exceptions to this rule are:

- When the choice contains an acronym or abbreviation that is usually capitalized, use accepted capitalization.
- When the choice contains a proper noun that contains a capital letter other than the first letter, use the accepted capitalization.

These rules for capitalization apply only to English language text.

Do not number the choices.

CUA defines a set of standard action bar choices: File, Edit, View, Options, and Help. Any of these standard choices that are valid for the application should be provided in the action bar. The actions should appear in the following order:

PICTURE 75

If the standard action bar choices are not adequate for your application, you may define other action bar choices. Remember that each action bar choice must have an associated pull-down. Selecting a choice on the action bar always results in a pull-down; it does not perform an action directly. Try to place your defined action bar choices from left to right, according to how often they are used, with the most frequently used choice at the left. An exception to this rule would be the need to group related choices.

The last (farthest right) choice in the action bar must be *Help*, if Help is supported.

Each type of object should have an action bar that supports the actions for that object type. For example, if your application supports only one type of object, then the action bar would be the same for each panel in your application. Your application, however, may support multiple object types. For example, a transportation application may support both truck objects and airplane objects. In that case, all panels for truck objects would have the same action bar, containing actions that apply to trucks, while all panels for airplane objects would have the same action bar, containing actions that apply to airplanes.

3.5.3 Cursor Position in Action Bar

The cursor is positioned initially in the blank space immediately to the left of the first choice. The cursor, therefore, is positioned to select the first action bar choice. As users tab from one choice to another, the cursor is positioned in the blank space immediately to the left of each choice.

3.5.4 Action Bar Pull-Down Layout

Locate the action bar pull-down contiguous to the bottom of the action bar. Position the pull-down choices so that the choice entry field is directly below the first non-blank character of its action bar choice.

If this alignment is not possible because the pull-down is wider than the space allowed to display it, which might happen if the last choice in the action bar is close to the right edge of the screen, reposition the pull-down horizontally to make sure it is fully visible. To make sure users do not forget which action bar choice is associated with a pull-down that has been repositioned, display the action bar choice with selected emphasis. Figure 76 illustrates the location of the pull-down in relation to the action bar.

Pull-down choices are left-aligned.

Use a solid line border for a pull-down if solid line characters are available. This pull-down border is shown in Figure 76.

PICTURE 76

Figure 76. Action Bar Pull-Down Border

If solid lines are not available, use one of the following as the border of a pull-down:

- For the top border, use the existing hyphen characters (-) that form the separator line below the action bar, except change the hyphen character to a period character (.) at the top corners where the pull-down intersects the separator line. Use the vertical bar character (¦) for the sides and the hyphen character for the bottom, except use an apostrophe character (') at each corner on the bottom. This pull-down border is illustrated in Figure 77.

PICTURE 77

Figure 77. An Alternate Action Bar Pull-Down Border

- For the top border, use the existing hyphen characters (-) that form the separator line below the action bar, except change the hyphen character to a period character (.) at the top corners where the pull-down intersects the separator line. Use the colon character (:) along the sides, including the bottom line, and use the period character between the colons on the bottom line. This pull-down border is illustrated in Figure 78.

PICTURE 78

Figure 78. Another Alternate Action Bar Pull-Down Border

If the pull-down contains only one selection field, groups of logically related choices may be separated by a blank line.

If the pull-down contains more than one selection field, use a blank separator line between each field. In this case, do not use a blank separator within a field.

3.5.5 Action Bar Pull-Down Content

You may use both single-choice and multiple-choice selection fields in pull-downs. The rules for selection fields in "Selection Elements" in topic 3.3.2 apply to selection fields used in pull-downs.

Place an entry field next to the first choice of single-choice selection fields. Place an entry field next to each choice of multiple-choice selection fields.

Pull-downs may not contain selection lists.

Number choices in single-choice selection fields. The number is the first character of the choice text. Use the rules for numbering selection fields in "Numbering Choices" in topic 3.3.2.1.1.

Do not use instructions or a function key area in pull-downs.

Ellipses for Pull-Down Choices: Place an ellipsis (...) after each choice in a pull-down that results in a pop-up. Do not put any spaces between the dots or between the ellipsis and the choice words. For example:

4. Save as...

Pull-Down Function Key Assignments: The Cancel action should be supported in all pull-downs, even though pull-downs do not have a function key area showing F12=Cancel.

You may assign function keys to the choices in a pull-down. These function keys are called *accelerators*. Display function keys to the right of the pull-down choices with the first character of each left-aligned. You may use function keys F1 through F24 unless they are reserved, such as F1=Help. Function keys that are used as accelerators do not have to be displayed in the function key area.

Assigned function keys are always active, whether or not the pull-down is displayed. If a key is assigned to a pull-down choice, you should display it in the pull-down.

Note: If your application supports a keyboard that has 12 function keys, some function keys might have multiple actions assigned to them. Those function keys must not be displayed in pull-downs. For information about supporting keyboards that have 12 function keys, see "Support for Keyboards with 12 Function Keys" in topic 3.7.3.

When users press a key that is assigned to a choice, the associated action occurs, even if users have not switched to the action bar or displayed the pull-down containing the choice. If any pop-ups are associated with the choice, the first pop-up appears when users press the function key assigned to that choice.

If users press a function key that is assigned to an action, and that action requires a previous object selection but no object has been selected, display a message that tells users to select an object.

Standard Action Bar Pull-Downs: In addition to defining standard action bar choices, CUA also defines standard pull-down choices and accelerator keys for some of those standard pull-down choices. For descriptions, see "Standard Action Bar Pull-Downs" in topic 2.5.4.1.

Exit in Pull-Down: The *Exit* action should always be the last choice in the first (farthest left) pull-down. If you use the standard action bar choices, Exit will be the last action in the File pull-down.

For details about the Exit action, see "Exit" in topic 3.7.6.6.

PICTURE 79

Figure 79. Pull-Down with Ellipses. The pull-down for the action bar choice ***Help*** contains six choices, all of which include ellipses that indicate a pop-up will appear. The choices are not assigned accelerator keys because the help accelerator keys are only active in help panels.

PICTURE 80

Figure 80. Pull-Down Choices with Accelerator Keys. The choices ***Undo, Mark,*** and ***Unmark*** have their accelerators listed on the same line.

PICTURE 81

Figure 81. Pull-Down with Two Selection Fields. This pull-down shows two single-choice selection fields. The first selection field contains the choices ***All*** and ***Some***; users can select either one. The second selection field provides three choices, ***By name, By date,*** and ***By subject***; users can select one of the three.

3.5.6 Action Bar Emphasis

Selected emphasis is used in the action bar to provide visual feedback that an action bar choice has been selected. A selected action bar choice remains displayed with selected emphasis while its pull-down is visible.

Unavailable emphasis is never used for an action bar choice, even when all of its pull-down choices are unavailable. An action bar choice must always be available so that users can display its pull-down to request Help on the unavailable pull-down choices.

3.5.7 How Users Interact with the Action Bar and Pull-Downs

Following is a general description of what happens when users interact with an action bar and action bar pull-downs. Figure 82 illustrates the flow of the user interaction in a panel with an action bar. Specific rules for user interaction follow this general description and illustration.

To use the action bar, users first must move the cursor to the action bar from another area of the panel. This can be done by using the Switch to action bar (Actions) request. After the cursor is in the action bar or in an action bar pull-down, requesting Switch to action bar again returns the cursor to where it was located in the panel before the previous Switch to action bar was requested.

Users also can move the cursor to the action bar by pressing the arrow, Tab, or New line keys. On some nonprogrammable terminals, the Home key also moves the cursor to the action bar. None of these keys causes a host interrupt, so the system does not know where the cursor was positioned before the switch to the action bar.

Action bar choices are implicitly selected (point-and-select) choices. The cursor position indicates the selected choice.

Users press the arrow keys or the Tab key to move the cursor in the action bar. The arrow keys move the cursor one character at a time, and the Tab key moves it from choice to choice.

Users move the cursor and select choices in a pull-down the same way they do in other areas of the panel. When users press the Tab key, the cursor moves from left-to-right, top-to-bottom through the action bar choices or through pull-down entry fields. Fields in other panel areas are protected while pull-downs are displayed. Users, therefore, cannot tab to the other panel areas.

PICTURE 82

Figure 82. User Navigation in a Panel with an Action Bar

Subtopics
3.5.7.1 Rules for User Interaction
3.5.7.2 What Happens When an Available Pull-Down Choice is Selected
3.5.7.3 What Happens When an Unavailable Pull-Down Choice is Selected
3.5.7.4 What Happens if There is No Object for an Action to Act On

3.5.7.1 Rules for User Interaction

The following rules describe how users interact with the action bar and pull-downs.

Switching the Cursor to the Action Bar: Users can switch the cursor from another area of the panel to the action bar by:

Pressing the Actions (Switch to action bar) key. The cursor appears on the furthest left choice in the action bar.

Pressing the Tab key until the cursor appears on the leftmost choice in the action bar.

Pressing the arrow keys until the cursor moves into the action bar.

Pressing the New line key until the cursor moves to the line containing the action bar.

In some environments, pressing the Home key.

When users request the Switch to action bar action, a host interrupt occurs. Moving the cursor to the action bar with Switch to action bar, therefore, has the advantage of telling the system where the cursor was located when users requested the switch. Users, therefore, can be switched back to that previous cursor location when they return to the panel area by using the Switch to action bar action.

The Switch to action bar request does not remove the tabs or protect the fields in the work area or a command area when the cursor is in the action bar. The application, however, does remove the tabs and protect the work area fields when a pull-down or pop-up resulting from a pull-down appears.

If users move the cursor to the action bar by pressing a cursor movement key, such as a Tab or arrow key, the system cannot tell where the cursor was located before it was moved. When the action is completed and the cursor moves from the action bar back to the panel area, the cursor is positioned in a location determined by the application.

Moving the Cursor in the Action Bar: Users move the cursor from choice to choice in the action bar by pressing the Tab key. With the Tab key, the cursor is not restricted to the action bar and will continue to move from field to field through the panel and eventually return to the first choice in the action bar. The arrow keys move the cursor one character at a time in the action bar or any area outside of the action bar.

If users type over any character in an action bar choice, the application restores the text on the next host interrupt and ignores what users typed.

If a pull-down is displayed, cursor movement with the Tab key is restricted to the action bar choices and pull-down entry fields.

If the cursor is moved out of a pull-down using the arrow keys, the cursor is returned to the pull-down when users press the Enter key.

Switching Out of the Action Bar: Users switch out of the action bar by:

Pressing the Actions (Switch to action bar) key or the Cancel key. If users pressed the Actions key to move the cursor to the action bar, the cursor is returned to where it was located before the switch to the action bar.

Pressing the New line, Tab, or arrow keys. If users did not take an action that caused a pull-down or a pop-up to appear, all fields are still active when the cursor returns to the work area.

Selecting an Action Bar Choice to Request a Pull-Down: Users select an action bar choice with the cursor.

Users request that a pull-down appear by pressing the Enter key while the cursor is positioned on an action bar choice or on the blank before the choice.

If the cursor is in the action bar but it is not in a position to select an action bar choice, issue a message telling users to position the cursor correctly.

The pull-down is displayed with the cursor in the choice entry field that precedes the first choice in the pull-down.

Requesting Another Pull-down to Appear: Users can request that another pull-down appear by pressing the Tab or arrow keys to move the cursor to another action bar choice and then pressing Enter. Users also can return to the action bar from a pull-down by requesting the Cancel action in that pull-down.

If users request Cancel from a pull-down, the currently displayed pull-down disappears immediately. If, however, the cursor movement keys are used to leave a pull-down, the currently displayed pull-down is removed when users request another action bar choice. The newly selected pull-down then appears.

Canceling a Pull-Down and Returning to the Original Panel Area: If users switched to the action bar by pressing the Actions key, users can Cancel a pull-down and return to their previous location in the panel area by pressing the Actions key. The pull-down disappears and all areas other than the action bar are reactivated. Any selections users made in the pull-down are ignored; that is, no actions are started and the values remain as they were before selections were made.

Canceling a Pull-Down and Returning to the Action Bar: Users can press the New line, Tab, or arrow keys to move back to the action bar. However, the displayed pull-down does not disappear until a host interrupt occurs, such as when users request another action bar choice.

Canceling a Pop-Up: When users request Cancel from the first pop-up that results from a pull-down, the cursor returns to its original position in the underlying panel if users switched to the action bar using the Switch to action bar action. If Switch to action bar was not used, the cursor is positioned in a location determined by the application.

Moving the Cursor in a Pull-Down: The rules for moving the cursor within pull-downs are the same as those for the work area. Cursor movement with the Tab key is restricted to the action bar choices and pull-down entry fields.

If a pop-up appears, the Tab key moves the cursor from field to field within the pop-up. All other areas are protected.

Selecting Choices from a Pull-Down: Users can select choices from a pull-down in one of the following ways:

- Typing the choice number
- Typing the selection character (the slash, /, or a country-designated character) in one or more choice entry fields in a multiple-choice selection field.

3.5.7.2 What Happens When an Available Pull-Down Choice is Selected

When users select an available choice from a pull-down and press the Enter key, the following actions occur:

If the pull-down does not cause a pop-up to appear, the pull-down disappears. The application determines what action occurs, based on what users selected.

If a selected choice causes a pop-up to appear, the pull-down disappears and the pop-up appears.

If users request Help while the pull-down is visible, the pull-down remains displayed, as shown in Figure 125 in topic 3.10.3.5. (The continued display of the pull-down is an exception to what normally happens, which is that the pull-down disappears.)

3.5.7.3 What Happens When an Unavailable Pull-Down Choice is Selected

If an unavailable pull-down choice is selected, the pull-down remains visible and a message appears. (This is an exception to what normally happens when a pop-up appears from a pull-down, that is, that the pull-down disappears.) The message indicates that the choice is currently unavailable and that users may request Help for that choice. The help panel for the choice describes the conditions in which the choice is available and in which the choice is unavailable. The pull-down remains displayed after the message is removed.

3.5.7.4 What Happens if There is No Object for an Action to Act On

Part 2, "Application Models" in topic 2.0 describes the object-action approach to interface design as the best use of the action bar. With this approach, objects in the work area are selected by application default or directly by users. Then users select the appropriate action from an action bar pull-down. However, if users select a pull-down action that requires a previous object selection and no such object has been selected, present a message that tells users to select an object.

3.5.8 Action Bar and Pull-Down Interaction Example

This section describes typical user interaction techniques with the action bar and pull-downs. The panel shown in Figure 83 is the starting point for this example of user interaction techniques.

PICTURE 83

Figure 83. Action Bar Interaction Example Panel

PICTURE 84

Figure 84. Action Bar Pull-Down Interaction Example Panel

Users select the **Past 7 days** action from the **View** pull-down in the action bar through the following keystroke sequence:

1. Users complete the fields in the work area by entering **Baker, Charles** for the patient name, a 1 for **Status**, and a 1 for **Type of order**.

2. Users press the Actions key. The cursor appears in the space preceding the **File** choice.

3. Users move the cursor to the **View** choice by pressing the Tab key twice.

4. Users press the Enter key. The pull-down for **View** appears and the cursor is on the choice entry field in front of the **Since admission** choice. At this point, the starting panel in Figure 83 now looks like the panel in Figure 84.

5. Users type a *2* in the choice entry field to select the **Past 7 days** choice.

6. Users press the Enter key. The pull-down would disappear and a pop-up would appear showing the *care and treatment* items for the last seven days that are classified as *outstanding* for Charles Baker.

3.6 Chapter 11. Command Area

If users may want to process system commands without leaving your application, CUA recommends that your application provide a *command area* to allow users to request those actions directly. Application actions also may be supported through the command area, giving users an alternative to using the action bar and pull-downs.

Subtopics

3.6.1 Command Area Layout

A command area may be located in the primary window, or it may be a new panel in a pop-up.

Primary window -- If you provide a command area, CUA recommends that you make it part of the primary window so it is always available. Locate the command area immediately above the function key area.

Pop-up -- If you need to maximize the available space in the primary window and if you expect commands only to be used occasionally, as an application option, you may provide the command area in a dialog pop-up that is available on request. For more information about the layout and presentation of pop-ups, see Chapter 14, "Pop-Ups" in topic 3.9.

All command areas, whether displayed in the primary window or a pop-up, contain a field prompt and an entry field.

In applications created in English, the field prompt consists of the word *Command,* a blank space, and a right-pointing arrow consisting of three equal signs and the greater-than symbol. Do not use leader dots. For example,

```
Command ===> (Line one of entry field is here)
(Optional second line is here)
```

Or, you may put the word *Command* on a separate line. For example,

```
Command
===> (Line one of the entry field is here)
(Optional second line is here)
```

CUA recommends that you provide delimiters to visually indicate the length of the command entry field, as you would for any other entry field. Use the underscore attribute if it is available. For displays that do not support the underscore attribute, it is an application option to use the underscore character.

If a command area is in a pop-up, the pop-up contains at least a function key area and the command area entry field with the command field prompt. The panel title of the pop-up is *Command*. Instructions may be included also.

PICTURE 85

Figure 85. Panel with Command Area. Because this entry model panel occupies the full screen, there is enough space for a two-line command area. Users can move the cursor to the command area by using the Retrieve action (F9).

PICTURE 86

Figure 86. Command Area in Panel with Action Bar. This text subset panel has a one-line command area. Users can either select actions from an action bar pull-down or enter the actions directly by typing the actions in the form of commands into the command area. The F10 key switches users to and from the action bar. The F9 key moves the cursor to the command area and retrieves the previously issued command.

3.6.2 How Users Interact with a Command Area

Users can interact with the command area using four common actions: Enter, Command, Prompt, and Retrieve. Help should be available also for the command area. Refer to Appendix A, "Key Assignments" in topic APPENDIX1.1 for the specific function keys that are assigned to Command, Prompt, and Retrieve.

Subtopics
3.6.2.1 The Enter Action
3.6.2.2 The Command Action
3.6.2.3 The Prompt Action
3.6.2.4 The Retrieve Action

3.6.2.1 The Enter Action

The Enter action processes a command that has been typed into the command area of a primary window or a pop-up.

The following rules apply for the Enter action:

A command is processed when the Enter action is requested after a valid command has been entered into the command area of a primary window or a pop-up. If the command is valid, the command is processed even if the cursor is not in the command area when Enter is requested. A command is valid when it contains in proper sequence a supported command name plus all the parameters that are required for that command.

If the command area contains a valid command name but does not contain all the required parameters, or if the command area contains a valid command name with invalid parameters, a pop-up is displayed that helps users complete the command parameters. Parameters that were typed correctly into the command area by users should be displayed in their respective entry fields in the pop-up. If invalid parameters were typed into the command area, the pop-up should indicate which parameters are invalid and display a message telling users that the command contained invalid parameters. If a pop-up does not have enough space to display the command with its supporting parameters, a series of pop-ups may be used. When the pop-up dialog is completed, users press the Enter key to process the command.

If the command area contains a misspelled or unsupported command name and if users press the Enter key, display a message informing users of the condition.

After successfully processing a command, the application should clear the command area. If the Command area is located in a pop-up, the pop-up remains displayed until users request Cancel. This is an exception to the rule for pop-ups, which is that Enter removes a pop-up. Because pop-ups are modal, if the cursor is moved out of the pop-up before the pop-up is canceled, the cursor is returned to the pop-up when Enter is requested.

3.6.2.2 The Command Action

The Command action allows users to request a pop-up that contains a command area. The pop-up appears after users request the Command action by pressing the assigned key or by requesting a Command pull-down choice. The Command action is supported only in applications that provide the command area in a pop-up.

3.6.2.3 The Prompt Action

As an application option, you may allow users to request the Prompt action for commands when the cursor is in the command area. Note, however, that even though Prompt is available, *a prompt indicator does not follow the command area entry field*.

The following rules apply for the Prompt action:

When users request Prompt for the command area and no command has been entered, a prompt list pop-up for the command area is displayed, containing a list of the available commands. As an application option, users can tailor the list of commands using global search characters.

When users select a command from the prompt list pop-up and press the Enter key, a pop-up is displayed to help users complete that command.

If the command typed into the command area or selected from the prompt list pop-up has no additional parameters, the application displays a pop-up telling users that the command is complete and that they can press the Enter key to process the command. Requesting Prompt should never process a command immediately.

If users request Prompt and the command area contains a valid command name but required parameters are missing or invalid parameters have been typed, a pop-up is displayed that allows users to complete the command parameters. If users already typed valid parameters into the command area, those parameters should be displayed in their respective entry fields in the pop-up. If invalid parameters were entered, the pop-up should indicate which parameters are invalid and display a message telling users that they entered invalid parameters. If a pop-up does not have enough room for all the supported parameters, a series of pop-ups may be used. When the pop-up dialog is completed, users press the Enter key to process the command.

3.6.2.4 The Retrieve Action

Users request the Retrieve action to re-display the last command that was issued. The previous command appears in the command area entry field. Users can change the command, add parameters to it, or press the Enter key to reissue it.

The following rules apply for the Retrieve action:

- If the cursor is not in the command area when Retrieve is requested, the cursor is moved to the command area, even if there is no command to retrieve.

- Users can request Retrieve repeatedly to back up through the list of previous commands in last-in-first-out order. CUA recommends that users be able to retrieve a minimum of 10 commands.

3.6.3 Using Both a Command Area and the Action Bar

If your application supports both an action bar and a command area, make sure that the two interface elements are not at odds with each another. When functions are available through both the action bar and the command area, you should regard the command area as a fast-path way of accessing action bar functions.

Make sure that functions available from both the action bar and the command area are consistent. For example, an action to send a document might be available through an action bar pull-down or its associated pop-up and through a command area. If the action in the pull-down is *Send*, the command name for that action also should be *Send*. It would be wrong to call the same action *Send* in one place and *Transmit* in the other. Also, do not use the same name for different functions.

Usually, an action will be in an action bar pull-down and its parameters will be in subsequent pop-ups. In some cases, an action bar pull-down choice refers to a group of actions, rather than a single action, and the resulting pop-up will list the specific actions as choices. In that case, the names for the specific actions should be the same in the pop-up as the names required for the command area.

Whether users select an action from a pull-down or from a pop-up, the parameters for that action should be specified in one or more pop-ups. These parameters should correspond to equivalent command-language parameters.

The command parameters available from the action bar pull-down and subsequent pop-ups might be a subset of all the parameters that users can type into the command area.

Figure 87 is the starting point for a series of examples illustrating the relationship of the action bar to the command area.

PICTURE 87

Figure 87. Command Area with Pull-Down. The action bar pull-down choices, such as ***New*** and ***Send***, also may be command names that users could type into the command area. If users select the ***Send*** choice from the pull-down, the pop-up in Figure 88 appears.

PICTURE 88

Figure 88. Pop-up with Command Parameters. This pop-up appears after users select the ***Send*** choice from the pull-down in Figure 87. It contains parameters for the ***Send*** choice. Users might be allowed to type these same parameters into the command area. If users type ***Send*** into the command area entry field and press the Enter key, this pop-up would appear.

You may want to use the action bar to help users become familiar with how to type commands in the command area. You might do this for various reasons, such as:

- The command interface is the fast-path technique.
- Users need to know the full command form of an action specification to incorporate the command into macro files, batch files, or execs.
- The system has a natural-language command interface.

If you created a *command-learn mode*, users would interact with action bar pull-downs and pop-ups to piece together a command and its parameters for the command area. Figure 89 illustrates a command line that contains the command and parameters presented in the pull-down and pop-ups shown in Figure 87 and Figure 88.

PICTURE 89

Figure 89. Command Area with Command. Users type the command and its parameters in the command area. These are the same command and parameters as listed in the action bar pull-down and pop-ups.

Compare this approach with the action bar alternative illustrated in Figure 87 and Figure 88.

3.7 Chapter 12. Function Key Area

The *function key area* is the bottom area of a panel where available actions and their key assignments are listed. Some of the actions that appear in the function key area are *common actions*, actions defined by CUA that have common meanings in all applications. Other actions that may appear in the function key area are unique to specific applications and, therefore, are not *common* in all applications. Individual applications determine the meanings of the application-specific actions.

You must always define the contents of the function key area for each panel.

The CUA rules and guidelines in this chapter apply to applications that support keyboards with 12 function keys and applications that support keyboards with 24 function keys. On a keyboard that has 24 function keys, every common action specified by CUA can be assigned a unique function key. However, this is not possible on a keyboard that has only 12 function keys. Therefore, CUA has defined two techniques for supporting the common actions. These two techniques are described in "Support for Keyboards with 24 Function Keys" in topic 3.7.2 and in "Support for Keyboards with 12 Function Keys" in topic 3.7.3.

Note: Remember that action bar pull-downs do not have a function key area. However, they may contain accelerator keys, as described in "Pull-Down Function Key Assignments" in topic 3.5.5.

Subtopics

3.7.1 Function Key Area Layout

Place the function key area at the bottom of the panel below the message area and below the command area, if one exists.

The function key area stretches across the entire width of the panel. The actions are listed horizontally.

All choices are presented in a text form, such as *F1=Help.*

The function key area may be separated from the work area by a blank line. This separation is not required. The command area or message area also may serve as a boundary between the function key area and the rest of the panel.

3.7.2 Support for Keyboards with 24 Function Keys

Applications that support keyboards with 24 function keys have the ability to display in the function key area all the common actions that could be active at one time for a panel. CUA, therefore, recommends that users be given the option of setting the function key area to either of two forms: *display* or *no display*. Whichever form users choose will affect the function key area of every panel in the primary window. The function key area in pop-ups is not affected.

Display -- Shows the function key area.

No display -- Removes the function key area in the primary window, making the space available to the application. The key assignments are still in effect; they are just not displayed. The function key area actions also may be available as choices in pull-downs.

Display is the default form.

The Display keys action toggles the function key area between display and no display.

When the Function Key Area Is Turned Off and On: When users change the function key area from no display to display, expand the function key area upward from the bottom of the panel. If the panel has a command area and message area, these areas move up accordingly. You decide which other parts of the panel to overlay to provide the space for the function key area. Your panel design should take into account the space needed to display the function key area.

Subtopics
3.7.2.1 Function Key Area Content

3.7.2.1 Function Key Area Content

All common actions that might be valid on the current panel are displayed in the function key area.

Required for all application panels in the primary window if the action is supported on the panel:

- F1=Help
- F2=Display keys
- F3=Exit
- F4=Prompt
- F5=Refresh
- F6=
- F7=Backward
- F8=Forward
- F9=Retrieve/Command (F9=Retrieve if the command area is in the primary window; F9=Command if the command area will be provided in a pop-up.)
- F10=Actions
- F11=
- F12=Cancel
- F13=
- F14=
- F15=
- F16=Mark
- F17=Unmark
- F18=
- F19=Left
- F20=Right
- F21=
- F22=
- F23=Undo
- F24=.

Required for all pop-ups:

- F1=Help
- F12=Cancel.

Required for all help panels whether full-screen or in pop-ups, if the action is supported in the panel:

- F1=Help
- F2=Extended help (except extended help panels)
- F3=Exit
- F4=
- F5=Tutorial (if a tutorial is provided)
- F6=
- F7=Backward
- F8=Forward
- F9=Keys help (except Keys help panels)
- F10=
- F11=Help index (except Help index panels)
- F12=Cancel.

F1, F3, and F12 are *reserved* by CUA and cannot be used for application-defined actions, even if the application panel does not support the CUA common actions assigned to these keys.

Any function key other than F1, F3, or F12 is *conditional* and may be used for any application-defined action, if the application panel does not support the CUA common action assigned to that key.

All numbered function keys, including your application-defined keys, should be shown in numeric order from left to right, starting on the first line if more than one line is used.

PICTURE 90

Figure 90. Displayed Function Key Area. This typical function key area contains four common actions, ***Help***, ***Exit***, ***Retrieve***, and ***Cancel***. It also contains an application-defined action, ***Add record.***

PICTURE 91

Figure 91. No Display of Function Key Area. This is the same panel as Figure 90, except that the no-display form was chosen for the function key area. The function key area actions are still available, even when the no-display form is in effect.

PICTURE 92

Figure 92. Function Key Area for a Panel in a Pop-Up

PICTURE 93

Figure 93. Function Key Area Used with Action Bar. The ***Switch to action bar*** action is provided by the Actions key (***F10***). ***Prompt*** is included to provide a list of patient names for the entry field.

3.7.3 Support for Keyboards with 12 Function Keys

To support the CUA-defined common actions on a keyboard that has only 12 function keys, some of the function keys have been assigned two common actions. (See "Function Key Area Content" in topic 3.7.3.1 for function key assignments.) Only 12 function keys can be supported by the application at one time. These 12 are displayed in the function key area and only the actions defined by these keys are active. To access the remaining actions, users press the F2 key, which displays the new definitions for the function key area, in addition to the function keys whose definitions have not changed. The F2 key toggles between the base set of function keys and the second set.

To avoid user confusion, function keys that have two meanings must not be displayed in pull-downs.

When your application supports more than one set of function keys, the function key area must always be displayed to keep users informed of the currently active actions. The Display keys common action, therefore, is not allowed when you support more than one set of function keys.

Applications that support only the first set of common actions are not required to support multiple sets of keys. CUA recommends, therefore, that the Display keys common action be supported so users can turn the function key area on and off. In this case, F2 is assigned to Display keys instead of to Set 1/Set 2. You can find the first set of common actions listed in "Function Key Area Content" in topic 3.7.3.1.

As an option, applications may support two or more sets of function keys. However, use this option cautiously. Users might become confused by too many sets of function keys. When supporting more than two sets, the F2 key cycles through the different sets.

Subtopics
3.7.3.1 Function Key Area Content

3.7.3.1 Function Key Area Content

Listed below are the two sets of common actions defined for function keys 1-12 in application panels.

Subtopics
3.7.3.1.1 Function Key Area Definition: SET 1
3.7.3.1.2 Function Key Area Definition: SET 2

3.7.3.1.1 Function Key Area Definition: SET 1

The following function key area common actions are required for all application panels in the primary window if the function is supported on the panel:

F1=Help
F2=Set 2
F3=Exit
F4=Prompt
F5=Refresh
F6=
F7=Backward
F8=Forward
F9=Retrieve/Command (F9=Retrieve if the command area is in the primary window; F9=Command if the command area will be provided in a pop-up.)
F10=Actions
F11=
F12=Cancel.

3.7.3.1.2 Function Key Area Definition: SET 2

The following function key area common actions are required for all application panels in the primary window if the function is supported on the panel (differences are noted in italics):

F1=Help
F2=Set 1
F3=Exit
F4=Mark
F5=Unmark
F6=
F7=Left
F8=Right
F9=Retrieve/Command (F9=Retrieve if the command area is in the primary window; F9=Command if the command area will be provided in a pop-up.)
F10=Actions
F11=Undo
F12=Cancel.

Help and pop-up function key definitions for keyboards with 12 function keys are the same as the definitions for keyboards with 24 function keys. See "Function Key Area Content" in topic 3.7.2.1 for these definitions.

F1, F3, and F12 are *reserved* by CUA and cannot be used for application-defined actions, even if the application panel does not support the CUA common actions assigned to these keys.

Any function key other than F1, F3, or F12 is *conditional* and may be used for any application-defined action, if the application panel does not support the CUA common action assigned to that key.

3.7.4 Action Abbreviations

You may abbreviate or shorten some, but not all, action names; if you do, use the following English abbreviations:

Action	**Abbreviation**
Backward	Bkwd
Display keys	Keys
Extended help	Ex help
Forward	Fwd
Help index	Index
Switch to action bar	Actions

You may not abbreviate any other defined actions.

Note: If your application will be translated into languages other than English, refer to your translation guidelines for possible restrictions on the use of abbreviations. For a list of publications about IBM* national language support, see Appendix D, "Recommended Readings" in topic APPENDIX1.4.

3.7.5 Engraved Keys

Keys that are engraved with a description of their function, such as Enter and Home, are not displayed in the function key area.

In some environments, actions that are usually assigned to F1 through F24 also may have engraved keys, for example, Help. If your application will be used only in those environments, you do not need to display the function keys assigned to those actions.

Even though you do not have to display F1=Help when keyboards in your application environment have an engraved Help key, you must still support F1 as an alternate help key.

If keyboards in your application environment have engraved keys, such as *Page* for scrolling actions, you do not have to display or support F7=Backward and F8=Forward. Be aware, however, that nonprogrammable terminals in the System/370* environments typically do not have engraved keys for scrolling actions.

3.7.6 Function Key Area Common Actions

Function key area common actions are actions that have common meaning in all applications. You assign those actions to keys so users can request the actions by pressing the appropriate keys. Some of the actions also can be requested from an action bar pull-down or by a command that users type into the command area.

Following is a description of each of the common actions defined by CUA. The key assignments for these actions are shown in Appendix A, "Key Assignments" in topic APPENDIX1.1.

Subtopics

3.7.6.1 Backward and Forward

The Backward and Forward actions must be provided for all panels that contain an area that can be scrolled vertically (up and down). For a description of these scrolling actions and their effect on a scrollable area, see Chapter 13, "Scrolling Panel Areas" in topic 3.8.

3.7.6.2 Cancel

Cancel allows users to back up in the dialog one panel at a time or to back up from a pull-down to the action bar. Specifically:

- Cancel must be supported but not displayed in all pull-downs.
- Include Cancel in all panels that are part of a sequence of panels that comprise a single unit of work.

Repeated Cancel requests let users back out of an individual function or an entire application panel-by-panel, until users reach the highest-level panel. At that point, another Cancel request has the same effect as the Exit action.

If the application determines that significant information could be lost because of the Cancel action, a confirmation message appears, prompting users to save or discard information.

Retaining Information: When users Cancel a panel, the information in the panel is either discarded or retained, depending on how you want to establish the default panel values for the next display of the panel. If the information is discarded and the panel is later re-displayed, the panel contains the default values set by the application.

If the information is retained and the panel is later re-displayed, the panel contains the same values as it did when users canceled the panel. That is, any changes users made to the default values are retained by the application while users back up in the dialog. These new values remain in place when users come forward again to the same panel. The panel reappears exactly as users last saw it.

Subtopics
3.7.6.2.1 Responses to Cancel

3.7.6.2.1 Responses to Cancel

Situation: A pop-up is displayed. Users request Cancel.

Response: The pop-up is removed.

Situation: A pop-up is displayed that resulted from another, still-displayed, pop-up that resulted from a pull-down. Users request Cancel.

Response: The second pop-up is removed and the dialog returns to the underlying pop-up.

Situation: Users request Cancel from a pop-up that resulted from a pull-down.

Response: The pop-up is removed and the dialog returns to the work area.

Situation: Users request Cancel from an action bar pull-down.

Response: The pull-down is removed and the dialog returns to the action bar.

Note: Figure 94, Figure 95, and Figure 96, illustrate the Set 1/Set 2 function key area for applications that support keyboards that have 12 function keys.

PICTURE 94

Figure 94. Cancel Action. Users moved the cursor to the ***Format*** choice in the action bar and pressed the Enter key. The ***Format*** pull-down is displayed. If users then request ***Cancel*** from the pull-down, the pull-down disappears and users resume the dialog with the action bar.

In Figure 94, if users select **Primary**, instead of requesting Cancel, a pop-up appears, as shown in Figure 95.

PICTURE 95

Figure 95. Requesting Cancel from a Pop-Up. This pop-up resulted from the ***Primary*** choice in the ***Format*** pull-down in Figure 94.

In Figure 95, three common actions are supported, **Enter** and the two actions shown in the function key area. **Cancel** terminates the entire dialog about formats, that is, the **Primary Format** pop-up, and returns users to the work area. In this pop-up, **Cancel** ignores any changes users made in the panel, so the information in the **Primary Format** pop-up remains unchanged.

If users press the Enter key, all selections and entries users made in the pop-up are processed. The pop-up is removed and the cursor is returned to the work area. If the format changes are applied immediately, the text is formatted based on the new information.

PICTURE 96

Figure 96. Cancel from a Second Pop-Up. An application might allow a second pop-up, like this one, to follow the pop-up in Figure 95. If users request ***Cancel*** in the ***Reformat Option*** pop-up, only that pop-up is removed and the dialog returns to the ***Primary Format*** pop-up.

3.7.6.3 Command

The Command action allows users to request a pop-up that contains a command area. The pop-up appears when users press the F9 key.

3.7.6.4 Display Keys

When users request the Display keys action, the function key area is turned on or off. The default is to display the function key area. When users first request Display keys, therefore, the function key area disappears. When users request Display keys again, the function key area reappears. For more information about the display of the function key area, see "Support for Keyboards with 24 Function Keys" in topic 3.7.2.

If the Display keys action is supported, it must be assigned to the F2 key. You also may allow users to request the Display keys action by selecting a pull-down choice or by entering a command.

The Display keys action is not allowed if any panel in your application supports more than one set of function keys.

3.7.6.5 Display Panel IDs

When users request the Display panel IDs action, the panel IDs are turned on or off. The recommended default is off.

You may allow users to turn the display of IDs on and off in several ways, such as by selecting a pull-down choice, by entering a command, or by pressing a key. CUA does not specify a key assignment. When this action is in a pull-down, the application determines the content of the pull-down.

3.7.6.6 Exit

Exit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. Repeated Exit requests return the dialog to the highest level in the panel hierarchy that users are aware of.

Exit is not used in pop-ups, except for help pop-ups, because Exit applies to the function or application as a whole, not just to the sequence of pop-ups. Users should Cancel a pop-up, return to the primary window, and Exit from there.

The following figures show the result of using the Exit action in different situations.

In Figure 97, users have finished editing a memo and want to exit the text editor. They have moved the cursor to the **File** choice in the action bar and pressed the Enter key, which resulted in the display of the File pull-down shown in Figure 97.

PICTURE 97

Figure 97. Exiting from an Action Bar Pull-Down

Users then select the Exit action and press the Enter key. Because significant information could be lost, this results in the display of the Exit pop-up, as shown in Figure 98. This pop-up gives users the option to leave the text editor and save any changes (**Save and exit**), to leave without saving changes (**Exit without saving**), or to return to the text editor to continue working on the memo (**F12=Cancel**).

PICTURE 98

Figure 98. Exit Pop-Up

The panel **Patient Services Records** in Figure 99, is one in a sequence of Patient Services panels that comprise the Patient Services function of the Medical Services application. If users press the Cancel key, the previous panel in the sequence of Patient Services panels is displayed. If, instead, users press the Exit key, the Patient Services function is ended and the dialog is returned to the Medical Services main menu, as shown in Figure 100.

PICTURE 99

Figure 99. Exit from a Function within an Application

PICTURE 100

Figure 100. Exit from an Application

If users then press the **Exit** key from the Medical Services main menu in Figure 100, the Medical Services application is ended and the dialog is returned to the panel that contains the menu of available applications, as shown in Figure 101.

PICTURE 101

Figure 101. Exit to the Highest Level

Figure 102 shows possible routes of navigation from panel to panel for a typical entry model application. The squares represent the panels. The scrolling actions Forward and Backward are included to show how movement within a panel relates to movement among panels.

PICTURE 102

Figure 102. An Entry Model Dialog Using the Enter, Cancel, and Exit Common Actions

Figure 103 shows possible routes of navigation for a text subset application.

PICTURE 103

Figure 103. A Text Subset Dialog Using the Enter, Cancel, and Exit Common Actions

3.7.6.7 Help

There are several help actions to assist users while they are using an application. The help actions are:

Help (F1): Provides information about a specific item or field, an application panel, or the help facility, as follows:

F1 provides *contextual help* when the cursor is on a choice or in an entry field in an application panel.
F1 provides information about the application panel, known as *Extended help*, when the cursor is not on a choice or in an entry field.
F1 provides information about the help facility when F1 is requested from a help panel.

Help is required in all application panels.

Extended help (F2): Provides information about the entire application panel from which users requested help.

An Extended help panel is required for every application panel.

Extended help is required in all help panels, except Extended help panels.

Keys help (F9): Provides a list of the application keys and their functions.

Keys help is recommended in all help panels, except Keys help panels.

Help index (F11): Provides an index of the help information available for the application. When users select a help topic from the index and press the Enter key, a help panel for that topic is displayed.

Help index is recommended in all help panels, except Help index panels.

Help for help (F1): Displays a description of how to use the help facility.

Provides information about how to use the help facility. Help for help is displayed when F1 is requested from any help panel.

Help for help is recommended in all help panels, except Help for help panels.

Tutorial (F5): Provides access to a tutorial, if one exists. It is an application option to provide a tutorial.

If a tutorial is provided, F5 is required in all help panels.

For more information about the types of help and the user interactions with help, see Chapter 15, "Help" in topic 3.10.

3.7.6.8 Left and Right

The *Left* and *Right* actions must be provided for all panels that contain an area that can be scrolled horizontally. For a description of these scrolling actions and their effect on a scrollable area, see Chapter 13, "Scrolling Panel Areas" in topic 3.8.

3.7.6.9 Mark and Unmark

The *Mark* action selects, or marks, the portion of text to be processed by a subsequent Cut, Copy, Paste, Clear, or Delete operation. If no text is currently selected, Mark selects and emphasizes the character at the current cursor position in the work area. If a character or portion of text is already selected, Mark selects and emphasizes all character positions from the previously selected portion through the current cursor position. The *Unmark* action removes the emphasis from the currently selected portion of text, thereby de-selecting it.

3.7.6.10 Prompt

When users request the Prompt action, a pop-up appears with information to help users complete an entry field. For a complete description of the Prompt action, see Chapter 9, "Prompt" in topic 3.4.

3.7.6.11 Refresh

When users request the Refresh action, the content of the current panel is affected in either or both of two ways. It may be restored to its original state, it may be refreshed to reflect the current status, or both actions may occur.

Situation: Users request Refresh while working on a panel after changing some default values.

> **Response:** The content of the panel is restored to its initial condition. That is, the entry field values are restored to their initial values and selection fields are restored to their default choices.

Situation: Users request Refresh while working on a panel that shows a *snap-shot* of information that is continually changing.

> **Response:** The content of the panel is updated to reflect the current status of the information. For example, if users delete file names from a panel that displays a list of file names and then request Refresh, the panel is re-displayed with the deleted file names removed. As an application option, Refresh may be automatic, if the system allows.

Situation: Users request Refresh while working on a panel that presents continually changing information and contains defaults.

> **Response:** One Refresh action performs both types of Refresh functions.

Situation: Users scroll to the middle of a scrollable action list, indicate selections (by typing action codes), and then request Refresh.

> **Response:** The selection indicators are cleared from the choice entry fields, but the current view of the list is retained. The list is not reset to the beginning.

3.7.6.12 Retrieve

Users request the Retrieve action to re-display the last command issued. If the cursor is not in the command area when Retrieve is requested, the cursor is moved to the command area, even if there is no command to retrieve. For more information, see "The Retrieve Action" in topic 3.6.2.4.

3.7.6.13 Set 1/Set 2

If your application supports more than one set of function keys, this action allows users to access the next set of function key definitions. If more than two sets of function keys are supported, the text that is used when displaying the function key definition is *Set* followed by the number of the next set. For example, the first set of function keys would contain:

```
  F2=Set 2
```

Sets of function keys are numbered consecutively, beginning with one. The last set of function keys would contain:

```
  F2=Set 1
```

3.7.6.14 Switch to Action Bar

The Switch to action bar action allows users to switch the cursor back and forth between the action bar and other panel areas. For more information, see "How Users Interact with the Action Bar and Pull-Downs" in topic 3.5.7.

3.7.6.15 Undo

The *Undo* action reverses the most recently executed user action. Because the Undo action deals with hidden objects, the Undo pull-down choice should be modified *dynamically* to reflect exactly what is being *undone*. For example, when the last action executed by users was *Cut* and users move the cursor to the Edit action bar choice and press the Enter key, users should see *Undo cut* as the first pull-down choice.

3.7.6.16 Enter

When users are finished interacting with a panel that contains entry fields, selection fields, a selection list, or an action list, the panel must be submitted to the application with a specific action request, such as an action selected from a pull-down or the Enter action. *Enter* tells the application to process the panel. Users request Enter by pressing the Enter key. Enter should always result in a visible response.

Although Enter is a common action, it is never displayed in the function key area. Because Enter is always required to complete a panel, it would be on every panel. Therefore, in order to save valuable screen space, Enter is not displayed.

Subtopics
3.7.6.16.1 Responses to Enter

3.7.6.16.1 Responses to Enter

Table 4 explains what happens when users press the Enter key in either text subset or entry model applications.

Table 4. What Happens When Users Press the Enter Key

Situation	**Response**
The command area is not empty.	Process the command area. See Chapter 11, "Command Area" in topic 3.6.
The command area is empty and the cursor is in the action bar.	Display the action bar pull-down. See Chapter 10, "Action Bar and Pull-Downs" in topic 3.5.
The command area is empty, the cursor is in the pull-down, and	
Users made valid selections.	Initiate the selected action.
Users made invalid selections.	Display the appropriate message.
Users selected an object-dependent action without first selecting objects.	Display the appropriate message. See Chapter 10, "Action Bar and Pull-Downs" in topic 3.5.
The command area is empty and the cursor is anywhere but in the action bar or pull-down, and	
Users have supplied all the required input and the input is valid.	Process the panel.
Users have supplied invalid input or only part of the required input.	Display a message and error emphasis, as appropriate.

3.8 Chapter 13. Scrolling Panel Areas

Users can scroll an entire panel or areas of a panel using either of two scrolling techniques: *cursor-independent scrolling* or *cursor-dependent scrolling*. This chapter explains the scrolling methods and the ways of indicating to users that an area is scrollable.

The following rules apply for both scrolling techniques:

A panel may have more than one scrollable area, as an application option. Put a separator between scrollable areas.

The action bar and pull-downs from the action bar are not scrollable.

Each scrollable area must have *scrolling arrows* that indicate to users the position of the information they are viewing in relation to the boundaries of the information. You may also place a visual indicator at the boundaries of the information. For example, if users scroll forward to the end of the information, an indication like *End of Data* may appear.

A panel area stops scrolling at the boundary of the information. When users reach this boundary, your application should de-activate panel area scrolling in that direction, so users will not go beyond the end of the information.

Scrollable information does not wrap.

Subtopics

3.8.1 Scrolling Actions

When there is information beyond the horizontal boundaries of the visible panel area, applications are required to provide vertical scrolling, so users can access the information beyond the visible boundaries. If all the information fits within the horizontal boundaries of the visible panel area, vertical scrolling is not required. The same applies for horizontal scrolling.

The scrolling actions are:

Backward -- displays information above the currently visible information in the panel area.

Forward -- displays information below the currently visible information in the panel area.

Left -- displays information to the left of the currently visible information in the panel area.

Right -- displays information to the right of the currently visible information in the panel area.

The Common User Access defines specific function key assignments for these four scrolling actions. You can find these assignments in Appendix A, "Key Assignments" in topic APPENDIX1.1.

The same four scrolling actions are used for both scrolling techniques. The techniques differ in the way the amount of scrolling is determined.

3.8.2 Scrolling Techniques

CUA defines two scrolling techniques: cursor-independent scrolling and cursor-dependent scrolling. Provide one of these techniques for each scrollable panel area or element.

Subtopics
3.8.2.1 Cursor-Independent Scrolling
3.8.2.2 Cursor-Dependent Scrolling

3.8.2.1 Cursor-Independent Scrolling

With cursor-independent scrolling, also known as *page scrolling*, the information is scrolled in fixed increments regardless of the position of the cursor when users request one of the scrolling actions. As an application option, you may keep the cursor stationary on the screen or in the information. If the cursor remains stationary in the information and if the choice it was on scrolls out of view, it is an application option to either stop the cursor at the boundary of the panel area or to move the cursor to the first visible choice or entry field in the panel area.

The *increment*, or amount, of information scrolled when users request one of the scrolling actions varies with the size of the visible area that is scrollable. CUA recommends you use the following defaults for scrolling increments for cursor-independent scrolling:

- The visible area minus one item (for forward and backward scrolling)
- The visible area minus one column (for left and right scrolling).

You also may use other scrolling increments, such as the full, visible area or a part of the visible area, for example, one-half or one-third of the visible area.

You determine what quantity constitutes an item or a column.

3.8.2.2 Cursor-Dependent Scrolling

With cursor-dependent scrolling, the initial position of the cursor determines the extent of the scrolling when users request one of the scrolling actions:

Backward -- repositions the information so that the item containing the cursor is at the bottom of the scrollable area.

Forward -- repositions the information so that the item containing the cursor is at the top of the scrollable area.

Left -- repositions the information so that the column containing the cursor is the farthest right column of the scrollable area.

Right -- repositions the information so that the column containing the cursor is the farthest left column of the scrollable area.

When the cursor is at a vertical or horizontal boundary and users request scrolling, cursor-independent scrolling (page scrolling) is performed.

You determine what quantity constitutes an *item* or a *column*, based on what is most appropriate for the environment in which scrolling is used. An item, for example, might be one line or a group of related lines. A column might be a single column of data or a group of related columns.

CUA recommends that cursor-dependent scrolling be provided when users might want to position a specific item at a vertical or horizontal boundary. For example, users might want to reposition a list of items so a specific item is at the top of the list, or they might want to reposition text in a text editor so a specific line is the first line.

3.8.3 Scrolling Information

Scrolling information tells users that more information exists outside the visible panel area, the position of the visible information in relation to the total amount of available information, and which direction to scroll the work area to display the unseen information. Scrolling information may appear in three forms:

Scrolling arrows -- recommended by CUA.

- Actual arrows are recommended, if available, in the following format:

  More:

- If actual arrows are not available, the following alternate characters may be used, in the following format:

  More: < - + >

Textual scrolling information -- may be used in combination with scrolling arrows.

Textual scrolling location information -- may be used in combination with scrolling arrows or with scrolling arrows and textual scrolling information.

Subtopics

3.8.3.1 Scrolling Arrows

Scrolling arrows indicate that additional information exists outside the visible panel area and shows users which direction to scroll to see that information.

Subtopics
3.8.3.1.1 Scrolling Arrows Location and Layout
3.8.3.1.2 How Users Interact with Scrolling Arrows

3.8.3.1.1 Scrolling Arrows Location and Layout

Locate scrolling arrows right-justified on a line above the panel area to which they apply and below the textual scrolling location information, if present. The format is:

```
    More:
```

If actual arrows cannot be displayed by some terminals, you may use the following alternate characters for the scrolling arrows, in the following format:

```
    More: < - + >
```

where,

< indicates there is information to the left of the visible area.
- indicates there is information above the visible area.
+ indicates there is information below the visible area.
> indicates there is information to the right of the visible area.

Lay out scrolling arrows with one space between the colon (:) and the first arrow position and one space between the arrows.

If the panel can be scrolled in all four directions, backward, forward, left, and right, reserve the entire space for scrolling arrows (the word More, the colon, and the four arrows or symbols), even if all the arrows or symbols are not currently used because users cannot currently scroll in all four directions. Maintain the space of the unused arrows or symbols with blanks. For example,

```
    More:
```

indicates the panel can scroll only right; however, the spaces for the missing three arrows are maintained because, at some time, users might be able to scroll the information in all four directions.

If the panel area can be scrolled in only two directions, you only have to reserve space for the scrolling arrows that represent those two directions. For example, if only forward and backward scrolling are potentially available,

```
    More:
```

indicates that the panel area can scroll only forward, but the space for the undisplayed Up arrow is maintained because, at some time, users might be able to scroll in that direction.

3.8.3.1.2 How Users Interact with Scrolling Arrows

Users press the Backward and Forward keys to view information above or below the information that is currently displayed. They press the Left and Right keys to view information that is to the left or right of the information that is currently displayed.

PICTURE 104

Figure 104. Scrolling Arrows. The scrolling arrows, ***More:***

indicate that more information is available above, below, and to the right of the information that is currently displayed. The **Bkwd**, **Fwd**, and **Right** actions can be used to bring the

3.8.3.2 Textual Scrolling Information

Textual scrolling information may be used in combination with scrolling arrows. Textual scrolling information is never used alone. *Bottom* is displayed below the scrollable area when users are viewing the end of the information. *More...* is displayed below the scrollable area when users can scroll forward.

Subtopics
3.8.3.2.1 Textual Scrolling Information Location and Layout
3.8.3.2.2 How Users Interact with Textual Scrolling Information

3.8.3.2.1 Textual Scrolling Information Location and Layout

Locate Bottom, or *More...* on the right side of the line below the scrollable area, using the following rules for determining which indicators to use:

If users can scroll forward, display *More...* on the right side of the line below the scrollable area. For example,

```
                              More:

    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|Right border of
    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|scrollable area
                                  More...
```

If users are viewing the end of the information, display *Bottom* on the right side of the line below the scrollable area. For example,

```
                              More:
    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|Right border of
    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|scrollable area
```

3.8.3.2.2 How Users Interact with Textual Scrolling Information

Users press the Backward and Forward keys, as indicated in the function key area, to view information above or below the information that is currently displayed.

PICTURE 105

Figure 105. Panel with Textual Scrolling Information. ***More...*** provides an additional indication that users can scroll forward.

3.8.3.3 Textual Scrolling Location Information

The optional *textual scrolling location information* is text with numbers that tells users the position of the visible information relative to the total amount of available information. For example,

```
    Lines   5 to  18 of 180
```

As an application option, you may include textual scrolling location information with scrolling arrows or with scrolling arrows and textual scrolling information for each panel area that scrolls. However, textual scrolling location information is never used alone.

Subtopics
3.8.3.3.1 Textual Scrolling Location Information Location and Layout
3.8.3.3.2 Textual Scrolling Location Information Content
3.8.3.3.3 How Users Interact with Textual Scrolling Location Information

3.8.3.3.1 Textual Scrolling Location Information Location and Layout

Place textual scrolling location information right-justified on a line above the scrolling arrows. It may appear on the same line as the panel title or top instructions. It also may appear on a blank separator line between the panel title or top instructions and the top of the scrollable area.

3.8.3.3.2 Textual Scrolling Location Information Content

Textual scrolling location information contains descriptive text in the following format:

    Wwwwwwwwww xx to yy of zz

where,

Wwwwwwwwww is the appropriate term for the format of the information items being displayed; for example, Pages, Lines, Entries, Documents, Letters, Memos, Parts, or Claims. Your application should supply a name that is appropriate.

xx to yy is the range of items being displayed. Optionally, **xx** may be an entry field into which users type the starting point for the range of information they want to see.

To yy is optional when no range is defined, such as **Page 1 of 4**.

of zz is the total number of information items in the list. It is omitted when the total is not known.

In designing your interface, keep in mind that your application might be translated to a language that would need to change the order of *xx, yy*, and *zz*.

You may use textual scrolling location information to describe the vertical and horizontal dimensions of a panel at the same time. For example,

```
Lines      5 to  18 of 254
Columns    1 to   5 of  40
```

3.8.3.3.3 How Users Interact with Textual Scrolling Location Information

As an application option, you may allow users to type over the first value (**xx**) in textual scrolling location information, as a way to reposition the list of items relative to the new starting point the users want. You may allow this by placing the value for **xx** in an entry field (*xx*). Textual scrolling location information with an entry field would look like the following:

Lines 5 to 18 of 254

If the value exceeds the range in either direction, the list is repositioned to the top or bottom, in the direction users requested.

Figure 106 shows a panel with scrolling arrows and textual scrolling location information.

PICTURE 106

Figure 106. Panel with Scrolling Arrows and Textual Scrolling Location Information. The Down arrow indicates that users can scroll forward. The textual scrolling location information, ***Lines 1 to 16 of 25***, is provided to give more specific details about the position of the visible information in relation to the beginning and end of the text.

3.8.4 Panel Area Scrolling Examples

This section illustrates two examples of panel area scrolling, one for cursor-dependent scrolling and another for cursor-independent scrolling. Both examples use a panel containing a scrollable list of documents.

Subtopics

3.8.4.1 Cursor-Dependent Scrolling

In the following example, consisting of three figures, the scrolling actions are based on these cursor-dependent scrolling rules:

The Forward scrolling action causes the line containing the cursor to be positioned at the top of the scrollable area.
The Backward scrolling action causes the line containing the cursor to be positioned at the bottom of the scrollable area.

Users request a list of documents and the panel in Figure 107 is displayed. Only the Down arrow is displayed in the scrolling arrows, indicating that users can scroll only forward.

PICTURE 107

Figure 107. Panel with a Scrollable Area

Users move the cursor to the line containing the document named **Monthly** and request the Forward action by pressing the F8 key (**F8=Fwd**). The result is shown in Figure 108. The document named **Monthly** is positioned at the top of the scrollable area. The Up and Down arrows are both displayed, indicating that users can scroll the panel area backward as well as forward.

PICTURE 108

Figure 108. Panel after Forward Scrolling Action

Users now move the cursor to the line containing the document named ***Reportjl*** and request the Backward action by pressing the F7 key (**F7=Bkwd**). The result is shown in Figure 109. The document named ***Reportjl*** is positioned at the bottom of the scrollable area. The Up and Down arrows are displayed, indicating that users can scroll the panel area both backward and forward.

PICTURE 109

Figure 109. Panel after Backward Scrolling Action

3.8.4.2 Cursor-Independent Scrolling

In the following example, consisting of two figures, the scrolling increment is the visible panel area minus one item.

PICTURE 110

Figure 110. Panel with Cursor-Independent Scrolling. This is the same panel as in Figure 107 in topic 3.8.4.1, but it is now set for cursor-independent scrolling. The Down arrow in the scrolling arrows indicates that users can scroll forward.

Users request the Forward action by pressing the F8 key. The result is shown in Figure 111. The position of the cursor does not affect the result.

PICTURE 111

Figure 111. Panel Scrolled to the Limit. The list in Figure 110 has been scrolled forward so that the item at the bottom of the scrollable area is now at the top of the area. All items that preceded **Reportau** in Figure 110 have been scrolled out of view. The Up arrow indicates that users are at the end of the list and can scroll only backward.

3.9 Chapter 14. Pop-Ups

A *pop-up* is an area of the screen enclosed by a border that extends the users' dialog with the panel in the primary window. A pop-up may occupy a portion of the screen or the entire screen. Pop-ups are associated with underlying panels and appear when the application wants to extend the dialog in the underlying panel. For example, a pop-up may be used to provide:

- Additional information, such as help or a message
- A command area
- An entry field, selection field, or selection list.

Your application may have a series of overlapping pop-ups associated with an underlying panel.

Note: The rules and guidelines for the positioning and layout of pop-ups apply only to pop-ups that are not full-screen.

Subtopics

3.9.1 Pop-Up Positioning

A pop-up is associated with an underlying panel, a pull-down, or another pop-up. You may extend a pop-up beyond the boundary of a pull-down or another pop-up that it is associated with.

You determine the position of a pop-up using two methods: *item-adjacent positioning*, or *offset positioning*.

Subtopics
3.9.1.1 Location of a Pop-Up by Item-Adjacent Positioning
3.9.1.2 Location of a Pop-Up by Offset Positioning

3.9.1.1 Location of a Pop-Up by Item-Adjacent Positioning

When a pop-up is related to an item in the underlying panel, pull-down, or other pop-up, position the pop-up in a way that overlays the least amount of relevant information.

PICTURE 112

Figure 112. Adjacent Positioning. The pop-up is positioned directly below the entry field to which it applies.

When an item for which a pop-up is presented is no longer visible, such as when the item was in a pull-down that was removed, position the pop-up, if you can, so that it is adjacent to where users were last focused. This technique minimizes user eye movement and, therefore, helps maintain visual continuity between the previous item and the pop-up.

PICTURE 113

Figure 113. A Pop-Up that Continues the Dialog in a Pull-Down. When a pop-up is displayed for a pull-down, the pull-down is removed and the pop-up is positioned adjacent to the previous position of the pull-down. For reference in this figure, the previous position of the pull-down is shown.

3.9.1.2 Location of a Pop-Up by Offset Positioning

If a pop-up is related to an underlying panel or another pop-up but not to a specific item or area on it, use offset positioning to locate the pop-up.

Vertically offset the pop-up below the title of the underlying panel or pop-up so the title is visible. Horizontally offset the pop-up to the right of the left boundary of the underlying panel or pop-up.

PICTURE 114

Figure 114. Pop-Up Displayed on Top of Another Pop-Up. The ***Reformat Option*** pop-up is positioned so the title of the underlying pop-up is visible.

3.9.2 Pop-Up Layout

Use a solid line border for pop-ups if line characters are available. For example,

PICTURE 115

Figure 115. Pop-Up Border

If solid lines are not available, use one of the following as the border:

A border composed of the hyphen character (-) across the top and bottom, the vertical bar character (¦) along the sides, a period character (.) at each corner on the top, and an apostrophe character (') at each corner on the bottom. This pop-up border is illustrated in Figure 116.

PICTURE 116

Figure 116. An Alternate Pop-Up Border

A border composed of the period character (.) across the top, the colon character (:) along the sides, including the bottom line, and the period character (.) between the colons on the bottom line. This pop-up border is illustrated in Figure 117.

PICTURE 117

Figure 117. Another Alternate Pop-Up Border

Make pop-ups large enough to display the information they must contain. Leave some blank space inside the pop-up border to set off the pop-up from the information in the underlying panel. If the information is too large to fit within a reasonably sized pop-up, make the pop-up scrollable.

3.9.3 Pop-Up Content

Pop-ups must contain a function key area. Pop-ups also may contain most types of panel elements (entry fields, selection fields, selection lists, action lists, protected text, a message area, or a command area), depending on your need to extend the dialog from an underlying panel.

3.9.4 Pop-Up Interaction

When a pop-up is displayed, the pop-up becomes active and the cursor is positioned in the pop-up.

Users must finish interacting with the pop-up before continuing with the dialog in the underlying panel or in another pop-up, unless the pop-up contains a help panel.

Subtopics

3.9.4.1 How Users Interact with Pop-Ups

3.9.4.1 How Users Interact with Pop-Ups

A pop-up usually appears as the result of an action taken by users. Users always can remove a pop-up by requesting the Cancel common action. Also, Help should be available for a pop-up.

The Enter action removes a pop-up unless the pop-up invokes another pop-up. In that case, when the last pop-up is completed, all pop-ups are removed.

If users move the cursor out of a pop-up using the arrow keys and press the Enter key or a function key, the cursor is returned to the pop-up. No action is processed.

Exit is not supported in pop-ups, except in help pop-ups.

3.10 Chapter 15. Help

While running an application, users occasionally require additional information about choices, fields, or how to proceed with the application. Providing help within your application allows users to easily and quickly access this information.

The purpose of help is to provide online assistance at the users' request. The help information should be sufficient to allow users to successfully complete their current task in the application, but not to provide extensive details or instruction about the application. The Common User Access recommends that you provide help for every application, according to the rules and guidelines in this chapter.

Subtopics

3.10.1 Types of Help Information

There are several types of help to assist users in completing their current task. When users request assistance, provide contextual help based on where the cursor is located. Also provide extended help (general information about the application panel). You should provide a listing of keys and their actions, an index of help topics, and information about how to use help. You may provide two other types of help: access to a tutorial and reference phrase help.

The application determines the content of each type of help in accordance with the following guidelines:

Contextual Help (F1): Specific information about the item the cursor is on. Contextual help is often referred to as *field* help. Contextual help information describes the purpose of the item and tells application users how to interact with that item.

Provide contextual help when users press the F1 key while the cursor is:

- On a choice in the action bar.
- In any selection field, selection list, or entry field. This includes selection fields in action bar pull-downs and in pop-ups. It also includes entry fields in the command area and in pop-ups.
- On a message.

If appropriate for your application, you may provide contextual help for other areas on a panel.

Figure 118 shows an example of contextual help for a selection field.

PICTURE 118

Figure 118. Contextual (Field) Help. Users moved the cursor to the choice ***Pharmacy*** and pressed the Help key (F1).

If the cursor is on an unavailable choice, CUA recommends that the resulting help information explain why a choice is unavailable and tell users the actions necessary to make it available. If the cursor is in an area of the screen for which no contextual help is available, such as a blank line, Extended help is provided.

Command Area Help: Two kinds of contextual help are available for the command area. If the cursor is in the command entry field and users press the F1 key without typing anything in the field or type an invalid command, provide information about the commands in the application. If users type a command and press the F1 key, provide information about that command.

Help is required in all application panels.

Extended Help (F2): Information about the contents of the application panel from which users requested Help. Extended help informs users about the tasks that can be performed on the application panel. It is specific to the panel but not to specific areas on the panel.

Extended help is displayed when users press the F2 key while the cursor is in any help panel, except an Extended help panel. Extended help is displayed also when users press the F1 key while the cursor is in an application panel but is not on a choice or in an entry field.

Extended help is required in all help panels, except Extended help panels. Provide an Extended help panel for every application panel.

Keys Help (F9): A list of the application keys and their assigned functions.

Keys help is recommended in all help panels, except for Keys help panels.

Help Index (F11): A list of the help information available for the application. When users select a topic in the index, the help information for that topic is displayed. Figure 119 shows an example of a help index.

PICTURE 119

Figure 119. Pop-Up with Help Index

Help index is recommended in all help panels, except Help index panels.

As an application option, you also may give users the capability to search the index for specific topics or categories of topics.

Help for Help (F1): Information about help. This information tells users how to get help and how to use the help facilities. For example, it tells users how to place the cursor in an entry field or on a choice, so they can get contextual help by pressing the F1 key. It also tells users how to get Extended help, Keys help, and the Help index from help panels.

Help for help is displayed when users press the F1 key while the cursor is in any help panel.

Help for help is recommended in all help panels, except Help for help panels.

Tutorial (F5): Access to a tutorial if the application provides one.

Providing a tutorial is an application option. Unlike other types of help, Tutorial help is intended to teach users about the application. The tutorial might, for example, include interactive examples or exercises. The application defines the structure and content of the tutorial.

If a tutorial is provided, F5 is required in all help panels.

Reference Phrase Help: Additional information about selected words and phrases within a help panel. Providing reference phrase help is an application option.

In a help panel, certain words and phrases may be identified for which additional information is available. These words and phrases are called *reference phrases*.

Users move the cursor to the desired reference phrase, using the arrow or Tab keys, and press the Enter key to display the associated information. Reference phrases are emphasized; see *Emphasized text* in Appendix B, "Color and Emphasis Table" in topic APPENDIX1.2.

Figure 120 shows an example of how reference phrases might be used to directly link related help information.

PICTURE 120

Figure 120. How Reference Phrases May be Used to Link Related Help Information. The help pop-up ***Data Base File - Help*** was displayed after users requested it from the help index. After the definition, there is a list of three related topics that users can select. You make each topic selectable by defining it as a reference phrase and by specifying the information to be displayed when users select the topic. If users move the cursor to the reference phrase ***Creating a data base file*** and press the Enter key, a help pop-up for that topic is displayed. Notice that the text of the referenced topic also contains two reference phrases, ***data description specifications (DDS)*** and ***interactive data definition utility (IDDU)***. These two reference phrases, which also can be selected by users, are linked to other information.

3.10.2 Help Pull-Down in Application Panels

If your application has an action bar, the pull-down from the Help choice must contain Extended help. The pull-down also should contain Keys help, Help index, and Help for help. As an application option, you also may provide access to a tutorial. CUA recommends that you provide also an *About...* choice so users can access the ownership and copyright information of the application. For more information about the About... choice, see Chapter 17, "Copyright Information" in topic 3.12.

If all help choices are provided, they must be shown in the following order:

PICTURE 121

Figure 121. Order of Help Choices

To maintain consistency, the help pull-down choices always should be assigned the choice numbers indicated in the help pull-down in Figure 121. If you do not provide all the help actions, you should leave one blank line between non-consecutive numbers.

Note that the Help action (contextual help) is not used in the action bar pull-down.

3.10.3 Help Panel Design

CUA specifies some special rules and considerations for the design of help panels.

Subtopics
3.10.3.1 Help Content
3.10.3.2 Help Title
3.10.3.3 Function Key Area in Help Panels
3.10.3.4 Translation Considerations
3.10.3.5 Help Panel Display

3.10.3.1 Help Content

Help panels typically contain only protected text. However, as an application option, you may:

- Use reference phrase help to provide more information about specific words and phrases in the help panel
- Present the help index as a selection list.

3.10.3.2 Help Title

Help panels should have a panel title that contains the word *Help* and identifies the item or panel that help is being provided for. For example, if an application panel has an entry field prompt *Number of copies*, the title of the help panel for that entry field might be:

    Number of Copies - Help

3.10.3.3 Function Key Area in Help Panels

The following function keys and assignments are required in all help panels, if the action is supported in the panel:

F1=Help
F2=Extended help (except Extended help panels)
F3=Exit
F4=
F5=Tutorial (if a tutorial is provided)
F6=
F7=Backward
F8=Forward
F9=Keys help (except Keys help panels)
F10=
F11=Help index (except Help index panels)
F12=Cancel.

If the application will be used on terminals that have an engraved Help key, the description *F1=Help* does not have to be shown in the function key area of the application panels and help panels, but F1 still must be supported in addition to the engraved Help key.

Except for F1 for contextual help, the key assignments for the Help common actions are reserved for help panels only; that is, the keys that are assigned to the Help common actions are active only when users are in help panels. Therefore, except for F1, the keys assigned to the Help actions may also be assigned to application actions and used in application panels unless they are already reserved by CUA for application panels, such as F12, which is reserved for Cancel.

3.10.3.4 Translation Considerations

To design an application that is suitable for translation to another national language, keep the length and layout of information in help panels and the sequence of the help index independent of the code. Provide contextual help based on what item the cursor is on, not on the screen coordinates of the cursor.

3.10.3.5 Help Panel Display

Display help in a pop-up if pop-ups are available. If pop-ups are not available, use full-screen help panels.

When the help pop-up is displayed initially, the cursor is placed in the help pop-up. Both the help pop-up and the underlying application panel are active. This allows users to interact with the underlying application panel by moving the cursor to the underlying panel with the arrow or Tab keys. For more information, see "How Users Interact with Help Pop-Ups" in topic 3.10.4.3.

If the size of the help pop-up does not allow display of all the help information, provide scrolling so users can view the rest of the information in that pop-up. When users request help from a help pop-up, try to display the new help in the same help pop-up; that is, the new help information replaces the information that was being displayed when users requested additional help. There might be circumstances, however, where it is more effective to use a different sized pop-up. For example, if a small amount of contextual help is being replaced by a large amount of other help information, such as a large help index, replacing the original pop-up with a larger help pop-up might be less obtrusive than forcing users to do repetitive scrolling actions to see all the information.

A help pop-up partially overlays the application panel from which it was requested. CUA recommends that you position the help pop-up to allow users to view and interact with the area of the underlying panel where the cursor was located when users requested Help.

When users request Help for an action bar pull-down, keep the pull-down displayed when you display the help pop-up. This is an exception to a rule for pull-downs, which is that a pull-down is removed when a pop-up is displayed as a result of requesting a pull-down choice.

PICTURE 122

Figure 122. Help in a Pop-Up. Users press the F1 key while the cursor is in the ***Output file*** entry field of this entry model panel. The help pop-up does not overlay the entry field. Users, therefore, can move the cursor to the underlying panel and type into the entry field without removing the help pop-up. Notice the contents of the function key area in the help panel.

PICTURE 123

Figure 123. Full-Screen Help Panel for System Command. This entry model help panel appears when users type the command name ***Erase*** in the command area of an application panel and press the F1 key.

PICTURE 124

Figure 124. Action Bar Pull-Down with Operating System Search Function

PICTURE 125

Figure 125. Help for a Choice in the Search Pull-Down. Notice that the Search pull-down remains displayed when the help pop-up is displayed.

3.10.4 Help Interaction

Requesting and ending help must not interfere with the users' current dialog in the application. When users end help, the application must display in the underlying panel all unchanged entries and selections that users made before they requested assistance as well as any changes users made while using help.

Subtopics

3.10.4.1 How Users Request Help Actions from the Action Bar

To request help from the action bar, users first move the cursor to the Help choice in the action bar and press the Enter key. The Help pull-down is displayed. Users then select a choice in the Help pull-down and press the Enter key. The requested information is displayed in a help pop-up.

3.10.4.2 How Users Request Help Actions Using Function Keys

Users request a Help action by pressing the function key assigned to the action, as listed in Appendix A, "Key Assignments" in topic APPENDIX1.1. After the request, users see a help panel containing the help information they requested.

Contextual Help and Extended Help: Application users get contextual help (help on a specific item) by positioning the cursor on the item for which they want contextual help and pressing the Help key (F1 or an engraved Help key), as indicated by path 1 in Figure 126.

The application users get Extended help by pressing the F1 key while the cursor is not in a contextual help area of an application panel (path 2 in Figure 126), or by pressing F2 from a contextual help panel (path 3 in Figure 126) or from any other help panel, except an Extended help panel.

If Extended help is requested from a contextual help panel, the Extended help panel replaces the contextual help panel.

If supported, Help for help is provided by pressing the F1 key from any help panel, except a Help for help panel (path 4 and path 5 in Figure 126).

PICTURE 126

Figure 126. How Users Get Contextual Help, Extended Help and Help for Help

Keys Help: If supported, application users get a list of the application keys and their functions by pressing F9 from any help panel except a Keys help panel, as indicated in Figure 127.

The Keys help panel replaces the help panel from which Keys help was requested.

PICTURE 127

Figure 127. How Users Get Keys Help from Contextual Help or Extended Help

Help Index: If supported, application users request the help index by pressing F11 from any help panel except a Help index panel, as shown in Figure 128. If users can search the index, the help panel from which users requested the index is replaced by a search panel (1 in Figure 128). The index panel with the search results (2) replaces the search panel. If a search capability is not available, the full index (3) replaces the help panel from which users requested the index. When users select a topic from the index and press the Enter key, a panel containing the requested information (4) replaces the index.

PICTURE 128

Figure 128. How Users Get the Help Index from Contextual Help or Extended Help. If the index is searchable 1, a panel is displayed that contains an entry field in which users may type one or more words describing the information desired. All topics are then searched and a list is displayed of those topics that match the word or words entered 2. If users do not type any words in the entry field, but simply press the Enter key, or if the index is not searchable 3, all the index topics are displayed. Users can scroll through the list to select the desired topic. When users select a topic and press the Enter key, information about the requested subject is displayed 4.

Figure 129 shows an example of users searching a help index.

PICTURE 129

Figure 129. Example of Users Searching a Help Index. In this entry model example, the help index is full-screen. Users press F11 on a contextual help, Extended help, or other help panel and are presented with an entry panel 1 that contains instructions

about how to search the index. When users type ***send message*** and press the Enter key, a single-choice selection list is presented, showing the results of the search. 2. To view a topic, users select a topic and press the Enter key. Notice that the displayed list in 2 contains some related topics that do not have either of the users' search words (***send*** or ***message***) in their titles.

Tutorial: If a tutorial is available, users request it by pressing the F5 key from any help panel.

Reference Phrase Help: To display the information associated with a reference phrase, users press the arrow or Tab keys to move the cursor to the reference phrase and then press the Enter key. The current content of the help pop-up is replaced by a help panel that contains information about the selected reference phrase. Figure 120 in topic 3.10.1 shows an example of reference phrase help.

To make it possible for users to tab to each reference phrase, the phrases must be defined as input fields. The usual underscore delimiter for an entry field is not used. Any changes to the field by users are ignored; that is, if users type over the reference phrase, restore the reference phrase the next time that help panel is displayed.

3.10.4.3 How Users Interact with Help Pop-Ups

When the help pop-up is initially displayed, the cursor is placed in the help pop-up. Both the help pop-up and the underlying application panel are active. This allows users to interact with the underlying panel by moving the cursor to the underlying panel with the arrow or Tab keys. Users can type information into any entry field or select choices from any selection field or list that is not obstructed fully or partially by the help pop-up. If the panel or pull-down for which Help was requested is removed, the help pop-up also is removed. Users can move the cursor back to the help pop-up at any time, using the arrow or Tab keys. While the cursor is in the help pop-up, only the help function keys are active. While the cursor is in the underlying panel, only the application function keys are active.

PICTURE 130

Figure 130. A Help Pop-Up for an Action Bar Pull-Down. When Help is requested for a pull-down choice, the pull-down remains displayed while the help pop-up is displayed. Notice the positioning of the help pop-up relative to the pull-down choice for which the help was requested. If users move the cursor back into the pull-down and press the Enter key while a choice is selected in the pull-down, the selected choice is processed and the pull-down and pop-up are removed.

3.10.4.4 How Users End Help

Users end help by requesting Exit or by repeatedly requesting Cancel to back out of one or more help panels. For example, the first time users request Cancel, the current help panel disappears and the previous help panel is re-displayed, if there is one. When only one help panel remains, Cancel ends help, the same as Exit does.

3.11 Chapter 16. Messages

Messages are feedback that tell users that something has happened because of a request they made.

Subtopics

3.11.1 Types of Messages

CUA defines three types of messages:

- Information
- Warning
- Action.

An *information message* tells users that a computer function is being performed normally or has been completed normally.

A *warning message* tells users that a potentially undesirable situation could occur. Users do not need to correct the condition immediately to continue. However, corrective action may be required later to avoid an error situation. See Figure 132 in topic 3.11.6 for an example of a warning message.

An *action message* tells users that an exception condition has occurred. Users must perform an action to correct the situation. Action messages are used in situations that range from minor application-related conditions that stop users from continuing with the current dialog to serious system-related conditions that stop users from continuing to work with any application in the system. Figure 133 in topic 3.11.6 shows an example of an action message.

3.11.2 Message Layout and Content

Messages may be displayed in two forms:

- A message area on the panel
- A message pop-up.

Message Area: Provide a message area on all panels. Locate it immediately above the command area. If there is no command area, locate the message area immediately above the function key area.

The message area in the panel may be any number of lines, as determined by your application. If the entire message will not fit in the number of lines allocated for the message area, truncate the message. Display the full text when users request Help for the message. If the users' first Help request on a message results in the display of the full message text, a second Help request displays help for the message.

Messages are left-justified in the message area.

Your application may allow users to perform some actions, such as scrolling, without removing the message. A help request should not cause the removal of a warning or action message.

Message Pop-up: Display a message panel in a pop-up if you need to provide a means *within* the message for users to respond to the message. For example, you may provide an entry field or a selection list. You also may display a pop-up message if it is especially important to get the users' attention.

If pop-ups are not available, use full-screen panels for messages that would normally appear in pop-ups.

The panels you create for message pop-ups may contain any combination of protected text, entry fields, selection fields, or selection lists. Arrange the elements in message pop-ups as you would in other panels.

Message pop-ups may have panel titles. If a panel title is used in a message pop-up, the title is the application name. A message pop-up may have a message area.

Message pop-ups that contain entry fields, selection fields, or selection lists must have *Cancel* in the function key area.

3.11.3 Message Removal

Follow these rules and guidelines for the removal of messages from the message area:

Remove an information or warning message when users take an action that the application can detect, such as pressing the Enter key or a function key, and when the message is no longer needed.

For example, when users press the Forward function key, the application can detect it. If an information message in the message area applies only to the currently visible part of the work area, the application could remove the message when it scrolls the panel. However, if the message applies to the work area in general, the application could leave the message in the message area as an aid to users as they scroll from one portion of the work area to another.

As another example, consider a warning message during the users' sign-on sequence that notifies users that their passwords will expire in ten days. The application could leave the message displayed during the entire sign-on process as a reminder. Or, the application could remove the message when users take a detectable action, on the assumption that a ten-day warning is not urgent enough to be persistent.

Remove an action message *only* when the application detects that users have corrected the condition that caused the message to be displayed.

Follow these rules for removal of message pop-ups:

When users press the Enter key or request the Cancel action in a message pop-up, remove the pop-up if it contains an information message or a warning or action message that does not allow input to the message pop-up.

If users try to continue the application without correcting the condition that caused an action message to be displayed, re-display the message until the condition is corrected.

Remove a warning or action message that allows input to the message pop-up when users request the Cancel action or supply any required input and press the Enter key.

3.11.4 Audible Feedback

On terminals capable of sounding a *beep*, a beep should accompany warning and action messages, unless users have turned off the beep.

You must allow users to turn on and off the beep that accompanies warning and action messages.

Try to avoid letting users encounter situations in which so many warning and action messages occur that the beeps become annoying.

3.11.5 Guidelines for Creating Messages

The usability of your applications depends to a large degree on how well users understand and respond to its messages. Consider these guidelines for creating usable messages:

Provide help for messages in pop-ups, as you would provide it for other panels.

If you provide in a manual additional help for messages, you may include an alphanumeric identifier in the online message, to help users locate the information in the manual.

In messages that indicate an error, tell users what is wrong and how to fix the problem.

Write messages and prompts in concise, complete sentences. Use short, simple words and the active voice. Avoid jargon, abbreviations, and acronyms.

Write messages in sentence-style capitalization. Use uppercase only for acronyms and for commands and macros whose syntax requires it.

Use the same terminology in all messages.

Emphasize any names defined by the system or supplied by users that subsequently appear in message text. For example,

> You have selected the "Reviews" file.

Issue an information message to tell users that an action has been completed if there is no other visible indication. For example,

> Search complete. String "443-X" not found.

Issue an information message to tell users that extended processing is delaying a system response, if there is no other visible indication. For example:

> Please wait.

When an action will take a long time, issue an information message to tell users that the action is partially complete. If possible, tell them how much has been done and what is still left to do. For example,

> 17 orders processed. 2 orders await processing.

If necessary, tell users what action to take to complete the request. For example,

> Type in your authorization code and press Enter to continue.

Avoid using *Yes* and *No* choices when asking users to confirm an action. Users might misinterpret what the message is asking. Use short phrases that describe the actions available. For example,

> Choose one.
> 1. Save and exit
> 2. Exit without saving

is explicit. While

> Do you want to exit without saving?
> 1. Yes
> 2. No

requires users to read the message carefully to interpret its meaning.

3.11.6 Message Pop-up Examples

PICTURE 131

Figure 131. Warning Message that May Require User Action. This warning message alerts users to a condition that is not necessarily an error and may not require immediate attention. The message is put in a pop-up to make sure that users see it. The last line of the message explains what users do to continue. A *beep* sounds when this message appears.

PICTURE 132

Figure 132. Warning Message. This message contains a selection field that allows users to confirm their intentions when a significant amount of information or time might be lost. The choices are ordered with the less damaging choice listed first, so it can be the default choice. The ***Cancel*** action removes the message and returns the dialog to the underlying panel. A beep sounds when this message appears.

PICTURE 133

Figure 133. Action Message with Entry Field. This message asks users to type more information. A beep sounds when it appears. ***Prompt*** is available for the entry field.

3.12 Chapter 17. Copyright Information

If your application is protected by a copyright, you should indicate that it is protected the first time users start your application. The Common User Access recommends one of the following techniques:

Write the copyright notice to the message area of the first panel displayed. Remove the copyright notice when users take an action that the application can detect, such as pressing the Enter key or a function key.

Provide the copyright notice as protected text in the initial panel of your application.

PICTURE 134

Figure 134. Panel with a Copyright Notice in the Message Area

If your application uses action bars, use the *About...* choice on the help pull-down to provide a pop-up that contains the following information about your application:

The name and version of the application
The application serial number, if any
The copyright notice of your company
The copyright notice of another company, if required
The Cancel action in the function key area so users can remove the pop-up.

For details about the format of the help pull-down, see "Help Pull-Down in Application Panels" in topic 3.10.2.

PICTURE 135

Figure 135. About Pop-Up. This pop-up shows the copyright and ownership information for a hypothetical application called ***XYZ Medical Services***. This pop-up is displayed when users select the ***About...*** choice in the Help pull-down.

APPENDIX1 Part 4. Appendixes

Subtopics

APPENDIX1.1 Appendix A. Key Assignments

You must assign keys to functions in your application using the rules and specifications in this appendix. The key assignments apply to the IBM* Enhanced Keyboard. See appropriate product documentation for key assignments for other keyboards, such as the IBM* Modifiable Keyboard.

This appendix also contains key assignments for personal computers emulating AS/400* and System/370* keyboards.

Subtopics

APPENDIX1.1.1 Rules and Guidelines for Assigning Keys

The following is a list of general rules and guidelines to help you assign keys:

Applications may use any keys not assigned by the Common User Access, including unshifted keys, and the Shift+key and Alt+key combinations, if the programmable workstation or nonprogrammable terminal allows applications to access these keys.

Avoid using any keys assigned by the operating system under which your application will run.

If your application is going to be used in a country other than the United States, do not assign functions to the alternate state (Alt+key) of the alphanumeric keys. Users, however, may assign functions to these keys, if you allow them to.

Use the Alt and Shift keys with other keys to change the meaning of the keys. The Alt and Shift keys are not meant to be used individually.

Do not reassign keys or make duplicate key assignments for the keys listed in this appendix, except as noted in the following tables.

If a function assigned to a function key is common to several applications, try to assign the function to the same function key in all applications.

If users press a key that has no assigned function in the current panel, nothing happens. A message may be issued indicating that an unassigned key was pressed.

CUA includes support for keyboards with only 12 function keys, promoting consistency with keyboards that have 24 keys.

The mapping of the 12-key configuration is based on the following:

Assignment of F2 for the mapping function. Pressing F2 remaps the function key area to the set of actions described in this appendix. Pressing F2 again returns the function key area to its initial set of actions or selects another configuration, as determined by the application.

Assignment of the same function key for important actions in the function key area for both 12- and 24-key keyboards. Those actions are Help, Set 1/Set 2, Exit, Switch to action bar, and Cancel.

Some actions are assigned different function keys in the 12-key and 24-key keyboards. They are Left, Right, Mark, Unmark, and Undo. Table 5 in topic APPENDIX1.1.2.1 and Table 6 in topic APPENDIX1.1.4.1 reflect this difference in the second and third columns by following the 12-key assignments and key numbers with a slash and the 24-key assignments. For example, in the column Key Engravings and Combinations, the key assignments for *Left* are F7/F19, where F7 is the assignment for the 12-key keyboard and F19 is the assignment for the 24-key keyboard. Similarly, in the Key Numbers column, the key numbers for Left are 118/44,57+118, where 118 is the key number for the 12-key keyboard and 44,57+118 is the number for the 24-key keyboard.

Following are some notes of caution:

Too many sets of function keys can be confusing to users. It is unlikely that entry model users will require alternate sets of function keys.

An action must always be assigned to the same function key, whether the action is in the function key area or is an accelerator in a pull-down.

If the Set 1/Set 2 actions are supported, the function key area will always be displayed. The Display keys action, therefore, is not required.

APPENDIX1.1.2 Key Assignment Tables

The tables in this section tell you what function to assign to what key or combination of keys. They list the functions and key assignments for the IBM* Enhanced Keyboard. Each key has a corresponding number on the keyboard layout so you can locate it. Look up the numbers you find in the table on the keyboard layout in Figure 136 in topic APPENDIX1.1.2.1.

Subtopics
APPENDIX1.1.2.1 How to Use the Key Assignment Tables

APPENDIX1.1.2.1 How to Use the Key Assignment Tables

Your application must adhere to the key assignments in the following tables if your application supports the functions listed.

F1, F3, and F12 are *reserved* by CUA and cannot be used for application-defined actions, even if the application panel does not support the CUA common actions assigned to these keys. Any function key other than F1, F3, or F12 is *conditional* and may be used for any application-defined action, if the application panel does not support the CUA common action assigned to that key. For example, if an application panel does not support Refresh, the application may assign F5 to an application-defined action.

In looking up key assignments in the key assignment tables, remember these few things:

A comma between numbers means two keys perform the same function. For example, the two Shift keys on the IBM* Enhanced Keyboard are numbered 44 and 57 on the keyboard layout table. Key combinations that include the Shift key are listed as 44, 57 and are followed by a plus sign (+) and a number.

The plus sign (+) designates combinations of two keys that must be pressed at the same time. For example, the *Backtab* function is performed by pressing the Shift key and the Backtab (|<--) key, so the number sequence for the Backtab function is 44,57+16, meaning that users press either Shift key (44 or 57) and the Backtab (|<--) key.

In the column Key Engravings and Combinations, the words in parentheses are the names of keys, not their engravings. These names are included for keys whose engravings are symbols, such as the arrow keys.

The dashes (--) mean the function is not available or is not assigned by CUA.

The key engravings and combinations are those on the United States layout of the IBM* Enhanced Keyboard, as shown in Figure 136.

Table 5. Key Assignments for IBM* Enhanced Keyboard

Function	**Key Engravings and Combinations**	**Key Numbers**
Backspace		
AS/400	<-- Backspace	15
System/370	<-- Backspace	15
Backtab		
AS/400	Shift+|<--	44,57+16
System/370	Shift+|<--	44,57+16
Backward (F-key conditional)		
AS/400	Page Up	85
System/370	F7	118
Cancel (F-key reserved)		
AS/400	F12	123
System/370	F12	123
Command (F-key conditional)		
AS/400	F9	120
System/370	F9	120
Delete (character)		
AS/400	Delete	76
System/370	Delete	76

Display keys (F-key conditional)		
AS/400	F2	113
System/370	F2	113
Down arrow		
AS/400		
(own	84	
System/370	arrow)	84
(own	arrow)	
Enter		
AS/400	Enter	64, 108
System/370	Enter	64
Erase to end of field		
AS/400	--	--
System/370	Erase EOF	81
Exit (F-key reserved)		
AS/400	F3	114
System/370	F3	114
First field on screen		
AS/400	--	--
System/370	Home	80
Forward (F-key conditional)		
AS/400	Page Down	86
System/370	F8	119
Help (F-key reserved)	*See* Note in topic APPENDIX	.112.1
AS/400	F1	125
	Help	112
System/370	F1	
Extended help		113
AS/400	F2	113
System/370	F2	
Help index		122
AS/400	F11	122
System/370	F11	
Keys help		120
AS/400	F9	120
System/370	F9	
Tutorial		116
AS/400	F5	116
System/370	F5	
Help for Help (F-key reserved)		112
AS/400	F1	112
System/370	F1	
Insert		
AS/400	Insert	75
System/370	Insert	75
Left (F-key conditional)	F7/F19	118/44,57+1 8
AS/400	F7/F19	118/44,57+1 8
System/370		
Left arrow		
AS/400	(Left	79
System/370	arrow)	79

	(Left arrow)		
Mark (F-key conditional)			
AS/400	F4/F16	115/44,57+1	5
System/370	F4/F16	115/44,57+1	5
New line			
AS/400	<-'	43	
System/370	<-' New Line	43	
Prompt (F-key conditional)			
AS/400	F4	115	
System/370	F4	115	
Refresh (F-key conditional)			
AS/400	F5	116	
System/370	F5	116	
Retrieve (command area) (F-key conditional)			
AS/400	F9	120	
System/370	F9	120	
Right (F-key conditional)			
AS/400	F8/F20	119/44,57+1	9
System/370	F8/F20	119/44,57+1	9
Right arrow			
AS/400	(Right arrow)	89	
System/370	(Right arrow)	89	
Set 1/Set 2 (F-key conditional)			
AS/400	F2	113	
System/370	F2	113	
Switch to action bar (F-key conditional)			
AS/400	F10	121	
System/370	F10	121	
Tab			
AS/400	-->|Tab	16, 106	
System/370	-->|Tab	16	
Undo (F-key conditional)			
AS/400	F11/F23	122/44,57+1	2
System/370	F11/F23	122/44,57+1	2
Unmark (F-key conditional)			
AS/400	F5/F17	116/44,57+1	6
System/370	F5/F17	116/44,57+1	6
Up arrow			
AS/400	(Up arrow)	83	
System/370	(Up arrow)	83	
Word left			
AS/400	--	--	

System/370	Alt+ (Left arrow)	60,62+79
Word right		
AS/400	--	--
System/370	Alt+ (Right arrow)	60,62+89

Note: Except for F1, you may assign the keys defined for help panels to other functions in panels not used for help.

PICTURE 136

Figure 136. IBM* Enhanced Keyboard, United States Layout. On personal

APPENDIX1.1.3 Keyboards Outside the United States

The IBM* Enhanced Keyboard has different key layouts outside the United States to accommodate double-byte character-set characters, right-to-left languages, and other national language differences. Figure 137 shows the base layout of the IBM* Enhanced keyboard.

PICTURE 137

Figure 137. IBM* Enhanced Keyboard, Base Layout

The differences for the United States keyboard and the two keyboards used outside the United States are as follows:

United States Keyboard

- Key numbers 14 and 15 combine to make a wide-key 15.
- Key numbers 42 and 43 combine to make a wide-key 43.
- Key numbers 44 and 45 combine to make a wide-key 44.
- Key numbers 56 and 57 combine to make a wide-key 57.
- Key numbers 131, 61, 132, and 133 combine to make the bar-key 61.

Japan Keyboard

- Key numbers 29 and 43 combine to make a dog-leg-shaped key 43.
- Key numbers 44 and 45 combine to make a wide-key 44.
- When Phonetic-to-Kanji conversion is not supported, key numbers 131, 61, 132, and 133 combine to make the bar-key 61.

Other Country Keyboard

- Key numbers 14 and 15 combine to make a wide-key 15.
- Key numbers 29 and 43 combine to make a dog-leg-shaped key 43.
- Key numbers 56 and 57 combine to make a wide-key 57.
- Key numbers 131, 61, 132, and 133 combine to make the bar-key 61.

Subtopics
APPENDIX1.1.3.1 Character Key Differences

APPENDIX1.1.3.1 Character Key Differences

Because CUA assigns functions to many function keys, you should be aware of some important differences in key layout when you design an application for character entry. The following table lists those keys for the United States keyboard, the Japan keyboard, and the keyboard used in other countries.

	Character Entry Keys Number
United States	1 to 57 excluding 14, 15, 16, 30, 42-45, 56-61. Plus the shifted state (keys 44,57+key) and the Alt state (keys 60,62+key) of these keys.
Japan	1 to 57 excluding 1, 15, 16, 29, 30, 43, 44, 45, 57, 58, 60, 61. Plus the shifted state (keys 44,57+key) and the Alt state (key 62+key) of these keys.
Other countries	1 to 57 excluding 14, 15, 16, 29, 30, 43, 44, 56-61. Plus the shifted state (keys 44,57+key) and the Alt state (keys 60,62+key) of these keys.

APPENDIX1.1.4 Emulator Key Mapping

This section contains tables listing key assignments for Common User Access functions when a personal computer emulator is used to access AS/400* and System/370* keyboard functions.

Subtopics
APPENDIX1.1.4.1 Support of AS/400* and System/370*
APPENDIX1.1.4.2 Personal Computer Key Assignments for Terminal Functions

APPENDIX1.1.4.1 Support of AS/400 and System/370**

Table 6. Key Assignments for Personal Computers Emulating AS/400* and System/370* Keyboards

Function	**Key Engravings and Combinations**	**Key Numbers**
Backspace		
AS/400 emulator	Backspace	15
System/370 emulator	Backspace	15
Backtab		
AS/400 emulator	Shift+|<--	Shift+44,57 16
System/370 emulator	Shift+|<--	Shift+44,57 16
Backward (F-key conditional)		
AS/400 emulator	Page Up	85
System/370 emulator	F7	118
Beginning of data		
AS/400 emulator	--	--
System/370 emulator	--	--
Beginning of field		
AS/400 emulator	--	--
System/370 emulator	--	--
Beginning of line		
AS/400 emulator	--	--
System/370 emulator	--	--
Cancel (F-key reserved)		
AS/400 emulator	F12	123
System/370 emulator	F12	123
Command (F-key conditional)		
AS/400 emulator	F9	120
System/370 emulator	F9	120
Delete (character)		
AS/400 emulator	Delete	76
System/370 emulator	Delete	76
Display keys (F-key conditional)		
AS/400 emulator	F2	113
System/370 emulator	F2	113
Down arrow		
AS/400 emulator (own	84	
System/370 emulator (own	arrow)	84
	arrow)	
End of data		
AS/400 emulator	--	--

System/370 emulator	--	--
End of field		
AS/400 emulator	--	--
System/370 emulator	--	--
End of line		
AS/400 emulator	End	81
System/370 emulator	--	--
Enter		
AS/400 emulator	<-' Enter	43
System/370 emulator	<-' Enter	43
Erase to end of field		
AS/400 emulator	--	--
System/370 emulator	End	81
Exit (F-key reserved)		
AS/400 emulator	F3	114
System/370 emulator	F3	114
First field on screen		
AS/400 emulator	--	--
System/370 emulator	Home	80
Forward (F-key conditional)		
AS/400 emulator	Page Down	86
System/370 emulator	F8	119
Help (F-key reserved)		
AS/400 emulator	F1	112
	Scroll Lock	125
	F1	112
System/370 emulator		
	F2	113
Extended help	F2	113
AS/400 emulator		
System/370 emulator	F11	122
	F11	122
Help index		
AS/400 emulator		
System/370 emulator	F9	120
	F9	120
Keys help		
AS/400 emulator	F5	116
System/370 emulator	F5	116
Tutorial		
AS/400 emulator	F1	112
System/370 emulator	F1	112
Help for help (F-key reserved)		
AS/400 emulator		
System/370 emulator		
Insert		
AS/400 emulator	Insert	75
System/370	Insert	75

emulator		
Insert/replace toggle AS/400 emulator System/370 emulator	Insert --	75 --
Left (F-key conditional) AS/400 emulator System/370 emulator	Shift+F7 Shift+F7	44,57+118 44,57+118
Left arrow AS/400 emulator System/370 emulator	(Left arrow) (Left arrow)	79 79
Mark (F-key conditional) AS/400 emulator System/370 emulator	Shift+F4 Shift+F4	44,57+115 44,57+115
New line AS/400 emulator System/370 emulator	Shift+<-'Enter Shift+<-'Enter	44,57+43 44,57+43
Prompt (F-key conditional) AS/400 emulator System/370 emulator	F4 F4	115 115
Refresh (F-key conditional) AS/400 emulator System/370 emulator	F5 F5	116 116
Retrieve (command area) (F-key conditional) AS/400 emulator System/370 emulator	F9 F9	120 120
Right (F-key conditional) AS/400 emulator System/370 emulator	Shift+F8 Shift+F8	44,57+119 44,57+119
Right arrow AS/400 emulator System/370 emulator	(Right arrow) (Right arrow)	89 89
Switch to action bar (F-key conditional) AS/400 emulator System/370 emulator	F10 F10	121 121
Tab AS/400 emulator	-->|Tab Shift+-->|Tab	16 44,57+106

System/370 emulator	-->|Tab	16
Undo (F-key conditional)		
AS/400 emulator	Shift+F11	44,57+122
System/370 emulator	Shift+F11	44,57+122
Unmark (F-key conditional)		
AS/400 emulator	Shift+F5	44,57+116
System/370 emulator	Shift+F5	44,57+116
Up arrow		
AS/400 emulator	(Up arrow)	83
System/370 emulator	(Up arrow)	83
Word left		
AS/400 emulator	--	--
System/370 emulator	Alt+ (Left arrow)	60,62+79
Word right		
AS/400 emulator	--	--
System/370	Alt+ (Right	60,62+89

APPENDIX1.1.4.2 Personal Computer Key Assignments for Terminal Functions

Table 7. Personal Computer Key Assignments for Terminal Functions

Function	**Key Engravings and Combinations**	**Key Numbers**
Attention		
AS/400 emulator	Esc	110
System/370 emulator	Esc	110
Clear		
AS/400 emulator	Pause	126
System/370 emulator	Pause	126
Command mode (text)	Alt	60
AS/400 emulator	Alt	62
	--	--
System/370 emulator		
Duplicate		
AS/400 emulator	Shift+Insert	44,57+75
System/370 emulator	Shift+Insert	44,57+75
Enter (additional)		
AS/400 emulator	Enter	108
System/370 emulator	Enter	108
Erase input		
AS/400 emulator	Alt+End	60,62+81
System/370 emulator	Alt+End	60,62+81
Fast left arrow		
AS/400 emulator	Alt+ (Left arrow)	60,62+79
System/370 emulator	Alt+ (Left arrow)	60,62+79
Fast right arrow		
AS/400 emulator	Alt+ (Right arrow)	60,62+89
System/370 emulator	Alt+ (Right arrow)	60,62+89
Field exit		
AS/400 emulator	Ctrl	64
System/370 emulator	--	--
Field mark		
AS/400 emulator	--	--
System/370 emulator	Shift+Home	44,57+80
Field- (Minus)		
AS/400 emulator	- (Minus)	105
System/370 emulator	--	--
Field+ (Plus)		
AS/400 emulator	+ (Plus)	106
System/370 emulator	--	--
Hex		
AS/400 emulator	Alt+F7	60,62+118
System/370 emulator	--	--

NumLock AS/400 emulator System/370 emulator	NumLock Shift+NumLock	90 44,57+90
Print screen (Personal Computer) AS/400 emulator System/370 emulator	Print Screen Print Screen	124 124
Print screen (AS/400 & System/370) AS/400 emulator System/370 emulator	Shift+Print Screen --	44,57+124 --
Reset AS/400 emulator System/370 emulator	Ctrl Ctrl	58 58
SysReq AS/400 emulator System/370 emulator	SysReq SysReq	60,62+124 60,62+124

APPENDIX1.2 Appendix B. Color and Emphasis Table

The following table lists the default colors and emphasis techniques you should assign to the specified panel elements. However, users should be allowed to change the color and selection emphasis for each element listed in the table.

Distributed applications that run on both AS/400* and System/370* may use the same color palette. AS/400* applications written to use a 3270 terminal may use either the As/400* or the System/370* color palette. However, using the System/370* color palette on AS/400* hardware is not recommended.

This table is only for nonprogrammable terminals. You can find the black and white palette colors for the programmable workstation Entry model in the *Programming Reference: Presentation Manager* in the *IBM Operating System/2 Version 1.2 Programming Tools and Information*, 6280212.

Subtopics

APPENDIX1.2.1 Colors and Emphasis for Nonprogrammable Terminals

APPENDIX1.2.1 Colors and Emphasis for Nonprogrammable Terminals

Table 8. Colors and Emphasis for Nonprogrammable Terminals

Panel Element	System/370 Dark Palette	System/370 Monochrome	AS/400 Dark Palette	AS/400 Monochrome
Background (Screen)	Dark	Dark	Dark	Dark
Action Bar and Pull-Downs				
Action bar	White	High	White	High
Un-selected choices	Yellow	Low	Green	Low
	Blue	Low	Blue	Low
Selected choices	Blue	Low	Blue	Low
Separator line	White	Low	White	Low
	Blue	Low and * over 1st character	Blue	Low and * over 1st character
Pull-down Border				
Available choices				
Unavailable choices				
Selected Emphasis	Yellow	High	Green	High
Work Area Elements	Blue	Low	Blue	Low
Panel ID	Blue	Low	Blue	Low
Panel title	Green	Low	Green	Low
Instruction	Blue	High	Blue	High
Column heading	Blue	High	Blue	High
Group heading	Green	Low	Green	Low
Field prompt				
Protected text	Green	Low	Green	Low
Normal text	Turquoise	Low	Green	Low
Variable output information	Turquoise	High	White	High
Emphasized text	White	Low	White	Low
	Blue	Low and * over 1st character	Blue	Low and * over 1st character
Selection Fields				
Available choices				
Unavailable choices	White	Low	White	Low
	Turquoise	Low	Green	Low
	Turquoise	Low	Blue	Low
Selection Lists				
Available choices	White	Low	White	Low
	Green	Low	Green	Low
Choice entry field	Turquoise	Low	Green	Low
	Turquoise	Low	Blue	Low
Normal input Underscore	Green	Low	Green	Low
Action lists	Turquoise	Low	Green	Low
List items	Turquoise	Low	Blue	Low
List item descriptions	White	High	White	High
	White	High	Blue	High
Action entry field	Green	Low	Green	Low
	Green	Low	Green	Low
Underscore				
Action codes and values				
Entry fields				
Normal input				
Underscore				
Emphasized input				
Underscore				

Descriptive text Prompt indicator (+)				
Panel Area Separator	Blank line	Blank line	Blank line	Blank line
Scrolling Information More: and arrows More..., Bottom Items x of...	White White Blue	High High Low	White White Blue	High High Low
Pop-up Window Border	Blue	Low	Blue	Low
Function Key Area Function keys	Blue	Low	Blue	Low
Cursor for single multiple choices	Depends on device type and cursor location			
Messages Information Text Border (if pop-up) Warning Text Border (if pop-up) Action Text Border (if pop-up)	 White White Yellow Yellow Red Red	 High High High High High High	 White White White White White White	 High High High High High High
Error Emphasis	Black on yellow	Reverse video	Reverse video	Reverse video

APPENDIX1.3 Appendix C. Designing an Object-Action Oriented Application

The application in "A Sample Text Subset Application" in topic 2.5.5 shows several aspects of object-action design in an application, but it cannot show everything. Therefore, some general guidelines and suggestions for designing object-action applications are offered here. Remember that the object-action design approach must be used for text subset applications and also can be used effectively in many entry model applications.

You must think about your application the way users would think about it. This might mean a reorientation of your user interface design instincts.

Traditional nonprogrammable terminal applications have used an action-object design, frequently using a hierarchy of menus or actions that implied an object or eventually led to selection of an object. To transform an action-object application into an object-action application usually requires a complete re-design of the interface, which might require a significant re-design of the application. CUA recommends that each new application be designed from the beginning to be object-action oriented.

Determine if users would benefit from the re-design of your application. Action-object might be a satisfactory approach for many traditional applications that have a limited number of actions. When you design a simple interface with only one or a limited number of actions, see Chapter 4, " Entry Model" in topic 2.6 for additional considerations.

This appendix guides you through the initial steps of design to determine the objects and actions that your application will support and to help you determine the appropriate design approach, object-action or action-object. If you conclude that object-action is the better approach, see Chapter 3, "Text Subset of the Graphical Model" in topic 2.5 for additional considerations.

This appendix contains some information about task analysis and usability testing. You can find references to additional information in Appendix D, "Recommended Readings" in topic APPENDIX1.4. A task analysis will help you identify typical user tasks. This will help you determine whether action-object or object-action is the better design approach for your application. Refer to "Understanding Users and Their Tasks" in Appendix D, "Recommended Readings" in topic APPENDIX1.4. That appendix also contains references about usability testing to help you validate the degree of usability in your application.

Subtopics
APPENDIX1.3.1 Objects
APPENDIX1.3.2 Actions
APPENDIX1.3.3 Testing

APPENDIX1.3.1 Objects

1. Determine the objects your users will work with and the sub-objects of those objects.

   It is essential that you question and observe potential users of your application to determine their perceptions of what they want to do with the application and how they expect to do it. As you make a list of the objects, use object names that users can identify with easily. Note the intended contents of each object.

   In the sample text subset application, the user (the department manager) needs to see and act upon the following kinds of objects: calendar, phone directory, personnel data, budget, and pending items. These objects are shown in Figure 24 in topic 2.5.5.

2. Examine the list of objects you just created to determine if users really need to see all of them to accomplish their tasks. Eliminate any objects that do not have to be visible to users. These are internal objects that can be implicit. For example, a directory record is an internal object that describes the location of the sub-objects that are used.

3. Analyze the list of objects you created to determine which objects users will work with and which objects are their sub-objects.

   In the sample text subset application, the object is the personnel data record of each employee in the department. This object is represented by the employee names in the Open dialog pop-up in Figure 26 in topic 2.5.5. The sub-objects are personal data, salary/performance, and awards. These sub-objects may be composed of other sub-objects, some of which are common to all objects. An example of such a sub-object in the sample text subset application is the employee name.

   The object is presented in the work area of a panel in the primary window, so users can modify it. This supports the principle of placing users in control, by making the interface visual.

4. Find a good physical model on which to base your application.

   The metaphor for the sample text subset application is the paper forms that the users are familiar with. When it is possible to define objects that your users will manipulate based on a model that they are already familiar with, you will make learning and using your application much simpler. Such a model is called a *metaphor*. Using metaphors is a technique that supports the principle of having users develop a *conceptual model* of the interface. The metaphor for the sample text subset application could be called an *employee's personnel records* metaphor.

   Note that basing the externals of your application on a metaphor is helpful only to the extent that there are parallels you can reinforce between what your application does and what happens in the metaphor you choose. When there is a close analogy and when users are familiar with the metaphor, users should learn and use your application more quickly.

5. Look at the bottom level of the objects you have organized. Ask yourself *again* if those really are helpful for users, or if they are internal objects.

   If you think they might be only internal, they probably are, so hide them from the users, if you can. If, as a result of higher-level actions, the actions that apply to these objects are implicit (or can and should be), the objects are probably internal objects.

6. Determine the best way to display the relationships among the objects so users can envision, select, and act on them.

   It might make sense to present just the different object types at the higher levels, rather than all the individual objects. This allows users to select first the type of objects they will be working with. For example, in the sample text subset application, the Open dialog pop-up in Figure 26 in topic 2.5.5 lists the employee names, rather than each available piece of the employees' personnel data (personal data, salary/performance records, and awards). Grouping objects into types might make sense also if there are a very large number of individual objects.

7. Determine the order for presenting objects.

   Consider the various ways to present the objects to users.

   More than one criterion might exist by which users will want to determine the order that the objects are displayed in. If that is true for your application, one of the actions you should support is the ability for users to specify the arrangement criteria.

After you have identified the objects in your application and their relationships with each other, and you have listed the contents of each object, identify potential users and conduct a paper test of the objects. If those users understand well the scope and intent of your application and understand the proposed contents of each object, you may proceed to refine the actions you have been considering for the objects.

APPENDIX1.3.2 Actions

The following steps are offered to help you determine the actions your application must support for its objects, the action bar categories and pull-down choices you will need for a text subset application, and the application commands, if you will provide a command interface to the actions. If you are designing an entry model application, you do not have to consider how to group actions and options in action bar categories and pull-downs, but you still need to follow the other considerations in this appendix. You will be repeating steps 2 through 7 for each object type you have defined.

1. Consider the top level of objects that your application will manage.

   Determine the categories of actions you want available for those whole objects when they are initially displayed. That is, when your application is started and it presents the set of objects it manages, determine the actions that can be applied to the entire object, rather than to the sub-objects.

   Some of these actions might be to create a new object of a particular type, to print, get information about, or save an object, or to save an object as another object.

   Another action is probably to open the object users have selected--to display its contents, and, therefore, to allow the finer-level actions that would be indicated by the action bar that is associated specifically with the contents of that object.

   If it is not already apparent, note the separation implied by this step. At any level of object hierarchy, the action bar categories, actions, and options apply to the objects shown, not to sub-objects of those objects. One of the applicable actions is *Open*. To work with the contents of one of the displayed objects, users would select the object and open it. That changes the focus of the display from the original objects to the contents of the opened object. At the same time, the available actions in the action bar and pull-downs become those that apply to the objects now displayed in the work area.

2. Pick one of the object types you have defined.

   Determine the actions you envision users performing on the different parts of this type of object. Make a list of the actions. For example, in the File pull-down in Figure 25 in topic 2.5.5 of the sample text subset application, you can see that **Open** and **Print** are the actions that can be performed on an employee's **Personnel data** record. In the Manage pull-down in Figure 30 in topic 2.5.5, you can see that three more actions can be performed: **Performance review**, **Recommend award**, and **Change salary/title**.

   When you name the actions:

   a. Use verbs, if possible.

   b. Use the same verbs for similar actions on different objects, if possible. The individual object types should convey any subtle differences.

      For example, if your application allows an agent to sign out a group of cases, a single case, or just an item from a case, such as a memo or a policy for update, it would be much better to use the same action name for that sign out function instead of a different action name for each.

Sign out item		Sign out item
Sign out case	would be much better than	Suspend case
Sign out group		Hold group

      As a result, it might seem that the same action can apply to many of the objects, with just slight differences in how the actions apply to the individual objects. That is all right. In fact, it is preferable to having actions named differently when, in fact, they act in a similar way.

      To the degree that an object type is similar to another object type, the set of actions available for each should also be similar, providing consistency within your application and across related applications. There may be some differences, of course, with some objects having more or fewer actions or options that

apply. But where the action, option, or category is similar between object types, the names used for each should be the same and the behavior of each should be similar.

If two object types are really different, their available actions may be different also.

If your application will support a command interface to application actions, the list you have just created should be the basis of the list of supported commands. The application commands should be supported in the same way as the corresponding actions available from the action bar.

3. Organize into related groups the actions you listed for one of your object types. These will become the action bar choices when that object is opened.

   The objects, actions, and their groupings are obviously very application-dependent. Only the designer, with the help of the intended users, human factors and usability personnel, and iterative testing by potential users, can determine the appropriate items and their categories.

   a. Look for actions that manipulate the object as a whole, such as Print, Save, and Send. List these actions under the category *File*. Include Exit; it always must be the last action listed in this first category.

   b. Look for other actions, ones that apply to sub-objects instead of to the entire object. For example, you might identify various editing actions. List those actions under the category *Edit*. You also might find actions that allow users to select different ways to look at the objects; list those under the category *View*. Other groupings may be for actions that *search*, *analyze* or *calculate* some or all portions of an object.

   c. Look for *options* that can be selected by users. These probably include most of the actions you listed that began with *Set*. Your instinct might be to show them in an action bar pull-down as *Set X* or *Set Y*, but they are normally shown as just the option name, such as *X* or *Y*, leaving the Set action implicit.

      The options you have listed might be numerous and might be better thought of as separate logical groups of options. If necessary, consider defining separate categories in the action bar, using the following guidelines:

      Look for a group of *formats* that can be selected for the object. For example, look for ways to display a column of numbers, such as with *$*, commas and decimals.

      List *fonts* in a separate group if they are available for selection.

      Also, there may be other groups of option collections, such as document *styles* or *layouts*.

      Treat all of these kinds of groups as you would the general *options* group described above.

      Notice that the names of these groups are most logically nouns; however, the choices in the associated pull-downs are still actions.

   d. If the application supports help, include a *Help* category, as described in Chapter 15, "Help" in topic 3.10.

   You now have an initial set of action bar choices and their respective pull-downs for this one object type, or a set of application commands for this object type.

4. Review the lists of actions and options you have categorized for this one object type.

   All the actions and options must have value for the users. If some of them are there because you thought they were nice or if they are there for your own use, do not include them.

   Eliminate extras that users cannot benefit from. Remember that, at

some point, the application becomes less usable because of the number of actions and options that users must consider. Determining that point of diminishing returns is very important.

5. Try to discover additional actions or options that might make your application easier to use.

   Try to anticipate options your users might find helpful. Although these options might not alter the final results produced by your application, they can greatly enhance the usability of your application.

   For example, users might want to change the format in which objects are presented. If that is true for your application, allow users to customize the format. For example, if your application contains columns, users might like to hide some columns from view or view columns not currently shown. They might like to add a column to make notes in, or they might want to control the size of each column.

   You can see another example in the sample text subset application in Figure 28 in topic 2.5.5, which shows a View pull-down. The View pull-down gives the department manager several ways to view the list of department employees. The view can show employees' **Personal data**, **Salary/Performance**, or **Awards**.

   At first, such options might seem like frills, but they might make the application users more productive. Note that such display options should be independent of how your application or system internally stores the related data. The way you organize the data that is to be displayed to users should be based on how users can best utilize the data. The way you work internally with data should be based on the efficiencies or constraints of the internal services used and should not be imposed on your users.

6. Logically group the categories that you have created.

   Do not mix actions and options.

7. Determine a reasonable number of actions or options per category.

   You should have at least two actions per category, that is, at least two choices in an action bar pull-down, but you should not have so many that users are confused.

   a. If you need fewer categories in the action bar and if the categories are related, consider combining small categories into one. You may use a separator in the action bar pull-down to separate each group of related actions from other groups in a single pull-down.

   b. Consider splitting large categories if there are logical subgroups within them and if you have room for additional action bar choices.

      You may want to split off additional categories of Set operations from the *options* category, as suggested above for formats, fonts, and styles.

   c. Occasionally, it also is desirable to use an item in a pull-down as a category of related actions. When this is done, selection of that pull-down action results in the display of a pop-up that shows a list of the actions users can choose.

      The nesting of truly different actions is not encouraged. If nesting is used at all, it is usually to select a related action or option that modifies the higher-level action, giving it a slightly different focus or meaning.

8. Repeat steps 2 through 7 for each of the object types you have identified.

9. Determine the dialog and prompt list pop-ups you will need.

   At this point you should design the dialog and prompt list pop-ups that are needed to help users complete the actions you have defined. It is important to remember that dialog pop-ups are an important vehicle for user input. Although they might be one of the last things you design, they are important. When you design any application-unique dialog pop-ups, be sure to follow the rules and

guidelines in Chapter 14, "Pop-Ups" in topic 3.9.

After you have created the categories of actions and options and have laid out the dialog and prompt list pop-ups, plan to test your entire proposal, using potential users.

APPENDIX1.3.3 Testing

Following are ***some*** aspects of your application design that you should test. For more information about testing, see Appendix D, "Recommended Readings" in topic APPENDIX1.4.

Initial testing may be done on paper. Once you are satisfied that you have an acceptable initial design, the subsequent iterations should be tested using prototype interfaces.

1. Your testing should try to identify the objects and actions your test users fail to find. There are several possible reasons for this happening. The object and action names might not have been intuitive. It also could be that the objects and actions users wanted to use were ones you had not envisioned. Or they could be there, but are placed in an unexpected part of the object organization or the pull-down categories.

2. Identify the objects and actions that the test subjects tried to use but abandoned because they could not figure out how to use them. For example, users chose the wrong object or action, chose it at the wrong time, or selected or used it in the wrong way.

   Add or change actions and objects as necessary, but with care. Remember that the test users have different backgrounds, so you may hear only from those whose viewpoints differ from what you have proposed. Those who found your proposed interface acceptable probably will not make specific comments.

3. Identify actions or objects that were provided but never used by the test users. This might have been because the test scenarios did not require them. Or users may have performed a task in some easier or more obvious way.

   Consider whether the unused actions and objects are really necessary and beneficial. Remember, however, that a short test cannot exercise all facets of a complex program. Also remember that in most test environments, most users are novice users. Therefore, unless the test is very long or unless some of the test subjects participate in many iterations of the test, they may not have the time or the initial inclination to exercise many options. Balance the test findings against longer-term user experience.

Plan to follow these design consideration procedures during several design iterations and at least once after each test cycle. For each test cycle, also consider how well your test results meet your acceptance criteria.

APPENDIX1.4 Appendix D. Recommended Readings

This bibliography lists selected publications that provide technical information on key principles, examples of user-centered design, and behaviorally oriented discussions of user interface technology and techniques. This set of references will help you get started with user interface design.

Subtopics

APPENDIX1.4.1 Getting Started

Baecker, R. and Buxton, W. (Eds.) *Readings in Human-Computer Interaction: A multi-disciplinary approach.* Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1987.

Brooks, A. P. *Introduction to SAA CUA Concepts Applicable to Mainframe Environments.* Technical Report 21.1268-R. Order from: IBM Corporation; MHV Library Learning Center; Dept. 65PA Mail Station 687; Kingston, NY 12401.

Heckel, P. *The Elements of Friendly Software Design.* New York: Warner Books, 1984.

Norman, D. *The Psychology of Everyday Things.* Hillsdale, NJ: Basic Books, Inc., 1988.

Rubenstein, R. and Hersch, H. *The human factor: Designing computer systems for people*. Massachusetts: Digital Press, 1984.

Shneiderman, B. *Designing the user interface*. Massachusetts: Addison-Wesley Publishing Company, 1987.

If your application is to be used in another country or translated into another language, the consistency provided by CUA must be reflected in the national language version of your application. You can ensure this consistency by designing your interface and application to support other countries and languages, from the moment you begin your design. Your application should adhere to this book and to the enabling rules and guidelines for national language support. In this book, discussions about where information is located on a panel are for left-to-right languages.

The following books contain the IBM* enabling rules and guidelines for national language support and implementation information:

National Language Information and Design Guide, Volume 1: Designing Enabled Products, Rules and Guidelines, SE09-8001

National Language Information and Design Guide, Volume 2: Left-to-Right Languages and Double-Byte Character Set Languages, SE09-8002

S/370 DBCS Design Guide, GG18-9095

S/370 DBCS Application Primer, GG18-9059.

APPENDIX1.4.2 Getting More Technical

These publications delve more deeply into specific topics in human-computer interaction.

Subtopics
APPENDIX1.4.2.1 User Interface Technology and Techniques
APPENDIX1.4.2.2 User-Centered Design: General Principles
APPENDIX1.4.2.3 User-Centered Design: Case Studies
APPENDIX1.4.2.4 Understanding Users and Their Tasks
APPENDIX1.4.2.5 Human Information Processing

APPENDIX1.4.2.1 User Interface Technology and Techniques

The references cited in the "Getting Started" section survey many user interface technologies and techniques. The Baecker and Buxton collection contains several important papers on user interface techniques. The references in this section cover key technologies and techniques in more depth. Note in particular the comprehensive Helander Handbook of Human-Computer Interaction.

Carroll, J., Smith-Kerker, P., Ford, J., and Mazur, S. "The minimal manual." *Human-Computer Interaction.* 3(2), 1987-1988. 123-154.

Doheny-Farina, S. (Ed.) *Effective documentation: What we have learned from research.* Cambridge, MA: The MIT Press, 1988.

Foley, J. D.; Wallace, V. L.; and Chan, P. "The human factors of computer graphics interaction techniques." *IEEE Computer Graphics and Applications,* November 1984: 13-48.

Helander, M. (Ed.) *Handbook of Human-Computer Interaction.* North-Holland: Elsevier Science Publishers, 1988.
(Very comprehensive collection covering nearly all major areas of human-computer interaction, including many chapters on specific user interface technology and techniques, aspects of task analysis, organizational impact of computer technology, software engineering tools, cognitive psychology, user-centered design and evaluation methods. The following list summarizes some chapters dealing with interface techniques: Billingsley, P. "Taking panes: Issues in the design of windowing systems." (Chapter 19); Elkerton, J. "On-line aiding for human-computer interfaces." (Chapter 16); Greenstein, J. and Arnaut, L. "Input devices." (Chapter 22); Ogden, W. "Using natural language interfaces." (Chapter 13); Smith, S. "Standards versus guidelines for designing user interface software." (Chapter 40); Tullis, T. "Screen design." (Chapter 18); Verplank. W. "Graphic challenges in designing object-oriented user interfaces." (Chapter 17);

Smith, D., Irby, C., Kimball, R. and Verplank, W. "Designing the Star user interface." *Byte.* 7(4), April 1983. 242-282.

APPENDIX1.4.2.2 User-Centered Design: General Principles

The following references provide an overview of key principles of user-centered design and usability engineering.

Gould, J. and Lewis, C. "Designing for usability: Key principles and what designers think." *Communications of the ACM.* 28(3), 1985, 300-311.

Helander, M. (Ed.) *Handbook of Human-Computer Interaction:* Gould, J. "How to design usable systems." (Chapter 35); Whiteside, J., Bennett, J., and Holtzblatt, K. "Usability engineering: Our experience and evolution." (Chapter 36).

Rubenstein, R. and Hersch, H. *The human factor: Designing computer systems for people* Massachusetts. Digital Press, 1984. (Cited earlier; in this context see especially chapters 2, 3, 4, and 11.)

APPENDIX1.4.2.3 User-Centered Design: Case Studies

A sampling of case studies in user-centered design, discussed in more depth.

Good, M., Whiteside, J., Wixon, D., and Jones, S. "Building a user-derived interface." *Communications of the ACM.* 27, 1985. 1032-1043.

Gould, J., Boies, S., Levy, S., Richards, J. and Schoonard, J. "The 1984 Olympic message system: A test of the behavioral principles of system design." *Communications of the ACM.* 30(9), September 1987. 758-769.

Percival, L. and Johnson, S. "Network management software usability test design and implementation." *IBM Systems Journal.* 25(1), 1986.

APPENDIX1.4.2.4 Understanding Users and Their Tasks

The following highly selective group of publications discusses users and task analysis at many levels, from computer interaction tasks to broad analyses of organizational change resulting from the introduction of technology. See also the discussions in references cited in the section "User-Centered Design: General Principles," such as the Helander handbook or Rubenstein and Hersh, "User-Centered Design: General Principles" in topic APPENDIX1.4.2.2.

Blackler, F. and Oborne, D. (Eds.) *Information technology and people: Designing for the future.* Leicester, UK: The British Psychological Society, 1987.
(This collection is wide-ranging, but it can point you to a large volume of literature outside the USA on task and work analysis.)

Brady, L. "User system analysis." In *Human factors in organizational design and management.* Eds. H. Hendrick and O. Brown. North-Holland: Elsevier Science Publishers, Inc., 1984.
(An example of a paper in a collection of short papers from a conference on the analysis of work and information technology.)

Bullen, C., Bennett, J., and Carlson, E. "A case study of office workstation use." *IBM Systems Journal.* 21(3), 1982.

Carroll, J., Mack, R., and Kellogg, W. "Interface metaphors and user interface design." Helander handbook, Chapter 3, 67-86.

Carroll, J. and Reitman Olson, J. "Mental models in human-computer interaction." Helander handbook, Chapter 2, 45-61.

Carroll, J. and Rosson, M. B. "Usability specifications as a tool in iterative development." In *Advances in human-computer interaction.* Ed. H. Hartson. Norwood, NJ: Ablex Publishing, 1985.
(Task analysis from the perspective of setting testable product objectives.)

Carroll, J. and Rosson, M. B. "The paradox of the active user." In *Interfacing thought: Cognitive aspects of human-computer interaction.* Ed. J. Carroll. Cambridge, MA: MIT Press, 1986.

Egan, D. "Individual differences in human-computer interaction." In *Handbook of Human-Computer Interaction.* Ed. M. Helander North-Holland: Elsevier Science Publishers, 1988. 543-568.

Gould, J., Boies, S., Levy, S., Richards, J. and Schoonard, J. "The 1984 Olympic message system: A test of the behavioral principles of system design." *Communications of the ACM.* 30(9), September 1987. 758-769.
(Cited above, relevant here for discussion of initial field work to understand users and their requirements.)

Lewis, C. and Norman, D. "Designing for error." In *User Centered System Design: New perspectives on human-computer interaction.* Eds. D. Norman and S. Draper. 411-432.

Norman, D. "Cognitive engineering." In *User Centered System Design: New perspectives on human-computer interaction.* Eds. D. Norman and S. Draper. Hillsdale, NJ: L. Erlbaum Associates, 1986. 31-61.

Percival, L. and Johnson, S. "Network management software usability test design and implementation." *IBM Systems Journal.*25(1), 1986.
(Cited above, relevant here for references to initial field work to understand users and their requirements.)

Potosnak, K. (Panel chair), Hayes, P., Rosson, M.B., Schneider, M., and Whiteside, J. "Classifying users: A hard look at some controversial issues." *Proceedings CHI'86 Human Factors in Computing Systems* (Boston, April 13-17, 1986), ACM, New York. 84-88.
(Brief panel discussion of issues; reference section useful.)

Regan, E. and O'Conner, B. *Automating the office: Office systems and end-user computing.* Chicago, IL: Science Research Associates, Inc., 1989.
(Textbook treatment, with focus on office automation; see especially chapters
17-22 on developing requirements, evaluating technology in the field.)

Reitman Olson, J. "Cognitive analysis of people's use of software."

In *Interfacing thought: Cognitive aspects of human-computer interaction.* Ed. J. Carroll. Cambridge, MA: MIT Press, 1986.
(Theoretical discussion of similarities and differences between analysis schemes for low-level computer-implemented tasks, and schemes for broader analysis of tasks in the workplace. References to academic literature on task and work analysis.)

Rockart, J. and Bullen, C. (Eds.) *The rise of managerial computing: The best of the Center for Information Systems Research.* Homewood, IL: Dow Jones-Irwin, 1986.
(Wide-ranging collection of papers on decision support tasks, users and requirements, including executive support. Lots of pointers into academic literature.)

APPENDIX1.4.2.5 Human Information Processing

The following publications provide comprehensive, technical overviews of psychological principles of human-computer and human-machine interaction, from traditional human performance perspectives to more cognitive interpretations.

Bailey, R. W. *Human performance engineering: a guide for system designers*. New Jersey: Prentice Hall, 1982.

Boff, K., Kaufman, L., and Thomas, J. *Handbook of perception and human-performance. Vol. 1, 2* New York: Wiley Publications, 1986.

Card, S. K.; Moran, T. P.; and Newell, A. *The psychology of human-computer interaction*. New York: Lawrence Erlbaum Associates, 1983.

Carroll, J. (Ed.) *Interfacing thought: Cognitive aspects of human-computer interaction.* Cambridge, MA: MIT Press, 1986.

Kantowitz, B. H. and Sorkin, R. D. *Human factors: understanding people-system relationships*. New York: John Wiley & Sons, 1983.

Nickerson, R.S. *Using computers: Human factors in information systems.* Cambridge, MA: MIT Press, 1986.

Norman, D. and Draper, S. (Eds.) *User Centered System Design: New perspectives on human-computer interaction.* Hillsdale, NJ: L. Erlbaum Associates, 1986.

APPENDIX1.5 Appendix E. Translated Terms

This appendix contains a list of user terms that are part of the Common User Access and that non-English-speaking application developers use in their native languages. These terms, shown in English and in translation, are presented to users in windows and pop-ups and in information about windows and pop-ups, such as messages, help, and documentation. Use only the designated translations for these terms.

In this appendix, the terms are translated into the following languages:

- Danish
- Dutch
- Finnish
- French
- French (Canadian)
- German
- Italian
- Japanese
- Norwegian
- Portuguese (Portugal)
- Spanish
- Swedish

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE
DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

DO NOT PRINT THIS PAGE

GLOSSARY Glossary

Note: Use entries designated *user term* in application windows and pop-ups and in information about windows and pop-ups, such as messages, help, and documentation. Do not use synonyms for these terms.

These definitions apply specifically to the nonprogrammable terminal environment.

+---+
¦ **A** ¦
+---+

About... (user term) A help action that displays ownership and copyright information about the application.

accelerator. A key or combination of keys that invokes an application-defined function. The user term is *function key*.

action. (user term) One of the defined tasks that an application performs. Actions modify the properties of an object or manipulate the object in some way.

action bar. (user term) The area at the top of the primary window that contains keywords that give users access to actions available in that window. After users request a choice in the action bar, a pull-down extension appears from the action bar.

action bar pull-down. (user term) An extension of the *action bar* that displays a list of available choices for a selected choice in the action bar. After users request a choice in the action bar, the pull-down appears. A pop-up can appear after a pull-down choice is requested.

action codes. (user term) Numbers or letters assigned to actions in an *action list*.

action entry field. The entry field used in action lists. Users can type an action code to invoke one of several possible actions against the object. Compare with *extendable action entry field.*

action-object. A process sequence in which users select an action and then select the objects to apply that action to. Contrast with *object-action*.

action list. (user term) A set of choices that allows users to select multiple objects and to specify that a different action be performed on each object at the same time.

action message. (user term) A message that tells users that an exception condition has occurred. Users must perform an action to correct the situation. Compare with *information message* and *warning message*.

Actions. (user term) See *switch to action bar.*

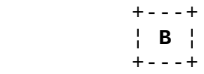
aligned. Arranged in a column in which all values either start in the same position (left-aligned) or end in the same position (right-aligned).

application. (user term) A collection of software components that users buy and install to perform specific types of work on a computer.

application option. A choice of appearance or interaction characteristics that programmers may implement in their application. If they decide to implement an application option, they must do so as described in this book. Compare with *user option.*

attribute byte. An undisplayed character that defines the characteristics of the field that follows it. On some terminals, some attribute bytes take up a position on the screen.

available choice. (user term) An item that the current state of the application allows users to select. Contrast with *unavailable choice.*

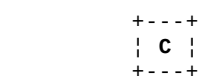


backspace. (user term) A typing action that moves the cursor to the left one space.

backtab. (user term) A typing action that moves the cursor to the previous entry field. The cursor moves from right-to-left and bottom-to-top. At the top, left field or choice, the cursor moves to the bottom, right field or choice. Contrast with *tab*.

Backward. (user term) A scrolling action that displays information above the information that is currently visible in a panel area.

border. (user term) A visual indication of the boundaries of a window or a pop-up.



Cancel. (user term) A common action that removes the current panel without processing it and returns the dialog to the previous panel in the hierarchy. Contrast with *exit*.

choice. (user term) An item that users may select. Choices appear in selection fields, selection lists, action lists, action bars, and action bar pull-downs. Text or a symbol can be used for choices.

choice entry field. An entry field used for selection fields and lists. Users type numbers, letters, or characters to indicate their selections.

Clear. (user term) An optional choice in the Edit pull-down that removes the selected portion from the object without copying it to a clipboard. The space is not compressed. Contrast with *delete*.

clipboard. (user term) The buffer that the system uses for temporary storage of data that is being manipulated.

column heading. (user term) One or more words at the top of a column of information that identify the information in that column.

Command. (user term) (1) A common action that is used to display a pop-up containing the command area. (2) The typed name and parameters associated with an action that can be performed by an application. A command is one form of action request.

command area. (user term) An area composed of two elements: a *command field prompt* and a *command entry field*.

command entry field. (user term) An entry field in which users type commands. The entry field is preceded by a *command field prompt*. These two elements make up the command area.

command field prompt. A field prompt showing the location of the command entry field in a panel (**Command ===>**).

common action. One of a set of actions that has common meaning across all applications, such as help, exit, and cancel. Common actions may appear in an action bar pull-down, the function key area, or both.

contextual help. (user term) A help action that provides specific information about the item the cursor is on. This help action is *contextual* because it provides information about a specific item as it is currently used. The information provided is specific to the meaning of the item within the application. Contextual help is also referred to as field-level help. Contrast with *extended help*.

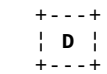
Copy. (user term) A choice in the Edit pull-down that produces a duplicate of the selected portion of an object on a clipboard without removing the selected portion from the object that is being edited.

cursor. (user term) A visual cue that shows users the current position of the keyboard input focus.

cursor-dependent scrolling. A method of scrolling panel areas that scrolls the information based on where the cursor is positioned when users request the scrolling action. The scrolling actions are *backward, forward, left,* and *right.*

cursor-independent scrolling. A method of scrolling panel areas that scrolls the information in fixed increments regardless of where the cursor is positioned when users request the scrolling action. The scrolling actions are *backward, forward, left,* and *right.*

Cut. (user term) A choice in the Edit pull-down that removes a selected portion from an object and copies it to the clipboard.



DBCS. See *double-byte character set*.

default choice. (user term) A selected choice provided by an application during the initial appearance of a selection field or list. Compare with *initial value*.

Delete. (user term) An optional choice in the Edit pull-down that removes the selected portion from the object without copying it to the clipboard. The space is compressed. Contrast with *clear*.

descriptive text. Optional text that appears to the right of an entry field. It provides additional information about the type of data that is required to complete an entry field.

dialog. The interaction between users and the computer.

dialog pop-up. A pop-up that is directly associated with a panel in a primary window. Through a dialog pop-up, users provide information needed to complete a dialog in the underlying panel.

Display keys. (user term) A common action that gives users the option to alternately turn off and back on the display of the function key area.

Display panel IDs. (user term) A common action that gives users the option to display *panel IDs*. The default is *off*.

double-byte character set (DBCS). A set of characters in which each character occupies two bytes. Languages, such as Japanese, Chinese, and Korean, that contain more symbols than can be represented by 256 code points, require double-byte character sets. Entering, displaying, and printing DBCS characters requires special hardware and software support.

+---+
¦ **E** ¦
+---+

Edit. (user term) A standard choice in the action bar that enables users to modify original documents and create new documents. It is used with a clipboard.

emphasis. Highlighting or color change that serves as visual cues to users. For example, an action bar choice is highlighted to show users a choice is selected.

Enter. (user term) A common action that submits information to the computer for processing. Enter tells the computer to perform selected actions on selected objects. Remember that *select* means to mark a choice; *Enter* means to send all designated choices to the computer for processing.

entry field. (user term) An area of a panel that users type information into. Compare with *selection field*.

error emphasis. A visual cue to users that they have typed incorrect information into an entry field.

Exit. (user term) A common action that ends a function or application and removes from the screen all windows associated with that function or application. Contrast with *cancel*.

explicit selection. A selection technique whereby users move the cursor to a choice and type a selection character to select that choice. Compare with *implicit selection.*

extendable action entry field. An entry field in an action list that allows users to type beyond the visible end of the field. Any subsequent text on the same line as the extendable action entry field is typed over.

Extended help. (user term) A help action that provides information about the contents of the application window from which users requested help. Contrast with *contextual help*.

+---+
¦ **F** ¦
+---+

fast path. (user term) A descriptive term referring to an alternate dialog interaction technique that is quicker than the usual interaction technique. For example, users can press function keys as accelerators to initiate an action immediately, instead of typing a command or accessing the action bar and a pull-down.

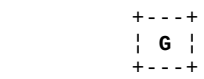
field prompt. Descriptive, static text that identifies selection fields, entry fields, and variable output information.

File. (user term) A standard choice in the action bar that enables users to work with stored objects.

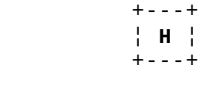
Forward. (user term) A scrolling action that displays information below the information that is currently visible in a panel area.

function key. (user term) A key that causes a specified sequence of application-defined operations to be performed when it is pressed. It is generally used to refer to keys labeled F*n*, for example, F1. The programmer term is *accelerator*.

function key area. (user term) The area at the bottom of a panel that contains function key assignments.



group heading. (user term) Word or words that identify a group of related fields.

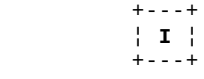


Help. (user term) (1) A common action that provides information about the item the cursor is on, an application panel, or the help facility. (2) An action bar choice that has an associated pull-down. Its pull-down contains choices that can be requested to invoke help actions.

Help for help. (user term) A help action that provides information to users about how the help facility works.

Help index. (user term) A help action that provides an index of the help information available for the application. A search capability is an application option.

help pop-up. (user term) A pop-up that contains information to assist users.



implicit selection. A selection technique whereby users move the cursor to a choice and that choice is automatically selected. Compare with *explicit selection.*

information message. (user term) A message that tells users that a function is performing normally or has completed normally. User acknowledgement may or may not be required, depending on the message. Compare with *action message* and *warning message*.

initial value. (user term) Information in an entry field that is provided by an application when an entry field is first displayed. An initial value can be a partial entry, such as the first few digits of a purchase order number, or a complete entry, such as a person's full name.

insert mode. (user term) A text-entry mode obtained by pressing the Insert key. Characters are inserted where the cursor is positioned. Text to the right is shifted to the right. Contrast with *replace mode*.

instructions. (user term) A panel element that tells users how to interact with a panel and how to continue with the application.

```
+---+
¦ J ¦
+---+
```

justified. Adjacent to the indicated margin. If attribute bytes are required, left-justified information appears one character position to the right of the left margin, with an unseen *attribute byte* in the first position. Compare with *aligned*.

```
+---+
¦ K ¦
+---+
```

Keys help. (user term) A help action that provides a listing of the application keys and their assigned functions.

```
+---+
¦ L ¦
+---+
```

Left. (user term) A scrolling action that displays information to the left of the information that is currently visible in a panel area.

left-aligned. See *aligned.*

left-justified. See *justified*.

```
+---+
¦ M ¦
+---+
```

Mark. (user term) An action in the Edit pull-down that selects a portion of an object to be processed by a subsequent Cut, Copy, Paste, Clear, or Delete operation.

message. (user term) Information not requested by users but shown by the application or system in response to user action or an internal process. Messages about status, exception conditions, or user actions should be distinguished from a "message" or note sent to users by other users over a communications link. See *action message*, *information message*, and *warning message*.

message pop-up. A pop-up used to display one of the defined message types. See *action message, information message*, and *warning message*.

More:. (user term) See *scrolling arrows.*

multiple-choice selection field. A type of selection field that allows users to select any number of choices or not make any selection. See *selection field*. Compare with *single-choice selection field.*

multiple-choice selection list. A type of selection list that allows users to select any number of choices or not make any selection. See *selection list*. Compare with *single-choice selection list*.

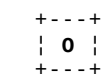
```
+---+
¦ N ¦
+---+
```

navigation. (1) The movement of the cursor around the screen using the keyboard. The user interface defines the key assignments for navigation. (2) Movement through an application using actions that are part of the user-computer dialog.

New. (user term) A choice in the File pull-down that allows users to create a new object.

new line. A cursor-movement function that moves the cursor to the first entry field on the next line that contains an entry field.

nonprogrammable terminal. A terminal attached to a host processor in which all or most of the user interface functions are controlled by the host.

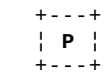


object. Anything users can manipulate as a single unit. Objects are the focus of the users' attention.

object-action. A process sequence in which users select an object and then select an action to apply to that object. Contrast with *action-object*.

Open.... (user term) A choice in the File pull-down that reads a stored object and displays it for users.

Options. (user term) A standard choice in the action bar that allows users to customize an application object. See *application option* and *user option*.



palette. (user term) A list of colors assigned to panel elements. Users can change the color of the individual elements within the palette.

panel. (user term) A particular arrangement of information grouped together for presentation to users in a window or pop-up. A panel is the application developer's mechanism for predefining the format of information to be presented to users in a window or pop-up. If the entire panel cannot be displayed at one time, users can scroll to see the rest of the information.

panel area. An area within a panel that contains related information. A panel area can be manipulated independently from the rest of the panel.

panel area separator. A solid, dashed, or blank line that provides users a visual distinction between two adjacent areas of a panel.

panel element. The smallest named part of a panel, such as *instructions, selection fields, entry fields,* and *panel titles.*

panel ID. (user term) An optional panel element located in the upper left corner of a panel that identifies that particular panel within the application.

panel title. (user term) A panel element that identifies the information in the panel.

parameter. (user term) A variable used in conjunction with a command.

Paste. (user term) A choice in the Edit pull-down that copies the contents of a clipboard into an object at a selected location.

pop-up. (user term) A bordered area of the screen that supplements the dialog that is occurring in the primary window. CUA requires or recommends support for four types of pop-ups. See *dialog pop-up, help pop-up, message pop-up*, and *prompt list pop-up*.

primary window. (user term) The window in which the main dialog takes place between the users and the application.

Print. (user term) A choice in the File pull-down that prepares an object for a printer and schedules it to be printed.

Prompt. (user term) A common action that users request while the cursor is in an entry field. Prompt produces a list of available choices for that entry field. Users can request a choice from the list and that choice will be inserted into the entry field.

prompt list pop-up. A pop-up associated with the Prompt action that contains a selection list of valid choices for the entry field from which users requested the Prompt action.

protected text. Information in a panel that users cannot interact with.

property. A unique characteristic of an object that can be changed. The properties of an object describe the object. Type style is an example of a property.

pull-down. (user term) See *action bar pull-down*.

reference phrase help. (user term) A help option that provides additional information about highlighted, application-defined words or phrases within a help panel.

Refresh. (user term) A common action that updates the content of the current panel, depending on the function of the panel and application. As an application option, Refresh may be automatic. The content of the panel can be restored to its initial condition or updated to reflect the current status of the information.

replace mode. (user term) A text-entry mode in which selected text is deleted and typed characters are inserted in its place. In replace mode, at least one character is always selected. Contrast with *insert mode*.

Retrieve. (user term) A common action that re-displays, one at a time, the previous commands that were issued. Each previous command is displayed in the command area entry field.

reverse video. A screen emphasis feature that interchanges both the foreground and background colors of an item.

Right. (user term) A scrolling action that displays information to the right of the information that is currently visible in the panel area.

right-aligned. See *aligned.*

right-justified. See *justified*.

```
+---+
¦ S ¦
+---+
```

Save. (user term) A choice in the File pull-down that writes an existing object into a storage device, such as a disk or diskette.

Save as.... (user term) A choice in the File pull-down that writes the existing object into a new object without changing the original.

screen. (user term) The physical surface of a terminal upon which information is shown to users.

scrolling action. A function that enables users to control the information visible in a window. The scrolling actions are: *backward, forward, left*, and *right*.

scrolling arrows. (user term) A type of *scrolling information* that consists of the word *More* followed by a colon and arrows or symbols that indicate the direction in which more information is available through scrolling.

scrolling increment. (user term) The fixed amount of information that scrolls, as determined by an application.

scrolling information. (user term) Any of the three types of information that appear in a panel to indicate the directions in which additional information is available by scrolling. See *scrolling arrows*, *textual scrolling information*, and *textual scrolling location information*

select. (user term) To mark or choose an item. Remember that *select* means to mark a choice; *Enter* means to send all selected choices to the computer for processing.

selected emphasis. A visual cue that shows users that a choice is currently selected.

selection field. (user term) A panel element that is not scrollable and contains a fixed number of choices. You should use this term in user documentation when referring to selection fields, selection lists, and action lists. See *single-choice selection field* and *multiple-choice selection field*. Compare with *entry field*.

selection indicator. A visual cue that shows users that a choice has been selected. Examples are the number of a choice and the slash character (/).

selection list. (user term) A set of choices that typically vary in content or number of choices and are potentially scrollable. See *single-choice selection list* and *multiple-choice selection list*.

single-choice selection field. A type of selection field from which users may select only one choice or not make a selection. See *selection field*. Compare with *multiple-choice selection field.*

single-choice selection list. A set of choices that are potentially scrollable and typically vary in content or number. They may also be fixed in content and number. Users may select only one choice or not make a selection. See *selection list*. Compare with *multiple-choice selection list*.

Switch to action bar. (user term) A common action that moves the cursor

to and from the action bar. It is listed as *actions* in the function key area.

```
+---+
¦ T ¦
+---+
```

tab. (user term) A typing action that moves the cursor to the next entry field. The cursor moves from left-to-right and top-to-bottom. At the bottom, right field, the cursor moves to the top, left field. Contrast with *backtab*.

textual scrolling information. A type of *scrolling information* that may be used with scrolling arrows. Textual scrolling information uses the words *Bottom*, and *More...* to tell users their relative position within scrollable information.

textual scrolling location information. A type of *scrolling information* that gives users optional information about their relative position within scrollable information. For example:

```
Lines 5 to 18 of 180
```

Tutorial. (user term) An optional help action that provides access to a tutorial if the application has one.

type over. (user term) An application option in action lists that allows users to change the characteristics of an object by typing over the currently displayed attributes of an object.

```
+---+
¦ U ¦
+---+
```

unavailable choice. (user term) An item that the application does not allow users to select. Contrast with *available choice*.

unavailable emphasis. A visual cue that shows users which choices are currently unavailable for selection.

underscore attribute. A display device feature that displays an underscore beneath each character position in an entry field to indicate the length of that entry field. Do not confuse with the *underscore character*, which is replaced when users type characters into an entry field.

Undo. (user term) A choice in the Edit pull-down that reverses the action of the most recently executed user action.

Unmark. (user term) An action in the Edit pull-down that de-selects a portion of an object that is marked to be processed by a subsequent Cut, Copy, Paste, Clear, or Delete operation.

user option. A choice of appearance or interaction characteristics that programmers give users during the operation of an application. Compare with *application option*.

```
+---+
¦ V ¦
+---+
```

View. (user term) A standard choice in the action bar that allows users to select different ways to look at an object without affecting the object itself.

warning message. (user term) A message that tells users that a potentially undesirable situation could occur. Users only need to respond to the message to continue. Corrective action may be required later to avoid an error situation. Compare with *action message* and *information message*.

window. (user term) An area of the screen with visible boundaries through which panel information is displayed. A window can be smaller than or equal in size to the screen. Windows can overlap on the screen and give the appearance of one window being on top of another.

work area. (user term) The part of a panel between the panel title separator and the message area. The work area is usually the main focus of user interaction with the panel.